(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
25 January 2001 (25.01.2001)

PCT

(10) International Publication Number
WO 01/06435 A1

(51) International Patent Classification⁷: G06F 17/60

(21) International Application Number: PCT/US00/19403

(22) International Filing Date: 14 July 2000 (14.07.2000)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
09/353,896 16 July 1999 (16.07.1999) US

(63) Related by continuation (CON) or continuation-in-part (CIP) to earlier application:
US 09/353,896 (CON)
Filed on 16 July 1999 (16.07.1999)

(71) Applicant *(for all designated States except US)*: E-DIALOG, INC. [US/US]; 1646 Massachusetts Avenue, Lexington, MA 02420 (US).

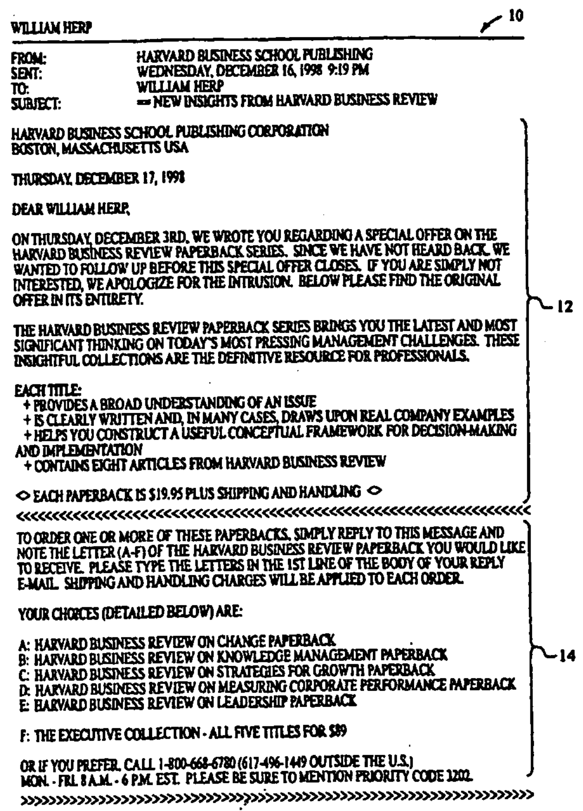
(72) Inventor; and
(75) Inventor/Applicant *(for US only)*: ESTES, Anthony, D. [US/US]; 4 Westmoreland Avenue, Arlington, MA 02174 (US).

(74) Agent: FEIGENBAUM, David, L.; Fish & Richardson P.C., 225 Franklin Street, Boston, MA 02110-2804 (US).

(81) Designated States *(national)*: AE, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, CA, CH, CN, CR, CU, CZ, DE, DK, DM, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, TZ, UA, UG, US, UZ, VN, YU, ZA, ZW.

[Continued on next page]

(54) Title: DIRECT RESPONSE E-MAIL



WILLIAM HERP

FROM: HARVARD BUSINESS SCHOOL PUBLISHING
SENT: WEDNESDAY, DECEMBER 16, 1998 9:19 PM
TO: WILLIAM HERP
SUBJECT: — NEW INSIGHTS FROM HARVARD BUSINESS REVIEW

HARVARD BUSINESS SCHOOL PUBLISHING CORPORATION
BOSTON, MASSACHUSETTS USA

THURSDAY, DECEMBER 17, 1998

DEAR WILLIAM HERP,

ON THURSDAY, DECEMBER 3RD, WE WROTE YOU REGARDING A SPECIAL OFFER ON THE HARVARD BUSINESS REVIEW PAPERBACK SERIES. SINCE WE HAVE NOT HEARD BACK, WE WANTED TO FOLLOW UP BEFORE THIS SPECIAL OFFER CLOSES. IF YOU ARE SIMPLY NOT INTERESTED, WE APOLOGIZE FOR THE INTRUSION. BELOW PLEASE FIND THE ORIGINAL OFFER IN ITS ENTIRETY.

THE HARVARD BUSINESS REVIEW PAPERBACK SERIES BRINGS YOU THE LATEST AND MOST SIGNIFICANT THINKING ON TODAY'S MOST PRESSING MANAGEMENT CHALLENGES. THESE INSIGHTFUL COLLECTIONS ARE THE DEFINITIVE RESOURCE FOR PROFESSIONALS.

EACH TITLE:
+ PROVIDES A BROAD UNDERSTANDING OF AN ISSUE
+ IS CLEARLY WRITTEN AND, IN MANY CASES, DRAWS UPON REAL COMPANY EXAMPLES
+ HELPS YOU CONSTRUCT A USEFUL CONCEPTUAL FRAMEWORK FOR DECISION-MAKING AND IMPLEMENTATION
+ CONTAINS EIGHT ARTICLES FROM HARVARD BUSINESS REVIEW

◇ EACH PAPERBACK IS $19.95 PLUS SHIPPING AND HANDLING ◇

<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<

TO ORDER ONE OR MORE OF THESE PAPERBACKS, SIMPLY REPLY TO THIS MESSAGE AND NOTE THE LETTER (A-F) OF THE HARVARD BUSINESS REVIEW PAPERBACK YOU WOULD LIKE TO RECEIVE. PLEASE TYPE THE LETTERS IN THE 1ST LINE OF THE BODY OF YOUR REPLY E-MAIL. SHIPPING AND HANDLING CHARGES WILL BE APPLIED TO EACH ORDER.

YOUR CHOICES (DETAILED BELOW) ARE:

A: HARVARD BUSINESS REVIEW ON CHANGE PAPERBACK
B: HARVARD BUSINESS REVIEW ON KNOWLEDGE MANAGEMENT PAPERBACK
C: HARVARD BUSINESS REVIEW ON STRATEGIES FOR GROWTH PAPERBACK
D: HARVARD BUSINESS REVIEW ON MEASURING CORPORATE PERFORMANCE PAPERBACK
E: HARVARD BUSINESS REVIEW ON LEADERSHIP PAPERBACK

F: THE EXECUTIVE COLLECTION - ALL FIVE TITLES FOR $89

OR IF YOU PREFER, CALL 1-800-668-6780 (617-496-1449 OUTSIDE THE U.S.)
MON. - FRI. 8 A.M. - 6 P.M. EST. PLEASE BE SURE TO MENTION PRIORITY CODE 1202.

>>

(57) Abstract: An e-mail message [10] is analyzed to derive response information concerning a commercial transaction. Based on the derived information, commercial transaction data are automatically generated in a format that is usable to complete the transaction.

WO 01/06435 A1

(84) **Designated States** *(regional)*: ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG).

Published:
— *With international search report.*

DIRECT RESPONSE E-MAIL

Field of invention

This invention relates to direct response e-mail.

Background of the Invention

In direct response e-mail, a vendor, for example, can sell a product to a customer by sending an e-mail message to the customer that describes the product and its price. The customer can order the product by returning an e-mail (sometimes called a direct response e-mail) that gives appropriate order information. The vendor can confirm the order by a return e-mail. The order information returned by the customer can sometimes be determined automatically using software that analyses the customer's reply e-mail.

Summary of the Invention

In general, in one aspect of the invention, an e-mail message is analyzed to derive response information concerning a commercial transaction. Based on the derived information, commercial transaction data are automatically generated in a format that is usable to automatically complete the commercial transaction.

In general, in another aspect of the invention, an e-mail message is sent to a customer offering a product or service for sale. The e-mail message includes locations for response by the customer to indicate his intention to order the product or service. The customer returns an e-mail message that includes the response. Based on the received e-mail, order information is automatically generated in a format usable automatically by an order fulfillment system to cause the order to be filled.

In general, another aspect of the invention includes automatically identifying response information which requires resolution of an issue with the source of the e-mail message and automatically managing an e-mail dialog with the source to resolve the issue.

In general, in another aspect, the invention features automatically sorting e-mail messages, based on response information contained in the messages, into e-mail messages that can be processed automatically to generate commercial transactions, e-mail messages in which the response information is inadequate to permit generation of commercial transactions, and e-mail messages that may be subjected to exception handling to yield information that is sufficient to generate commercial transactions.

In general, in another aspect, the invention features automatically generating a confirmatory e-mail message to the source of the e-mail message confirming that a commercial transaction has been or will be completed.

In general, in another aspect, the invention features receiving inbound e-mail messages that result from corresponding outbound e-mail messages associated with a marketing program, the inbound messages containing response information, each of the outbound messages being associated with a distinct piece of the marketing program. The response information in each of the inbound messages is automatically associated with the corresponding distinct piece of the marketing program.

In general, in another aspect, the invention features automatically merging response information with corresponding information in a database for use in completing transactions.

In general, in another aspect, the invention features identifying inbound e-mail messages that cannot be processed automatically to generate commercial transactions, and using the database information to assist in exception handling of the identified inbound messages.

Other advantages and features will become apparent from the following description and from the claims.

Brief Description of the Drawing

Figures 1A through 1C and 2A through 2B show e-mail messages.

Figure 3 is a block diagram of a direct response e-mail system.

Description of the Preferred Embodiments

Outbound e-mail messages

The two e-mail messages shown in figures 1A through 1C and 2A through 2B are examples of outbound messages associated with commercial transactions.

The example message 10 shown in figure 1 offers Harvard Business Review products. Message 10 includes basic copy 12 that is similar to basic direct marketing copy of the kind that is commonly used in e-mail marketing. Message 10 also contains a section 14 giving instructions on how to order the products.

Inbound e-mail messages

To take advantage of the offer shown in figure 1, the recipient creates a reply e-mail message (the direct response message) and types the letters of the items that he wants to order in the first line of the body of the message. In other examples, the letters could be typed in the subject line or the last line of the body of the message. The user is also asked to correct and complete shipping and e-mail address information that has been merged into the outbound e-mail message in a section 16. In section 16, each of the entries is bounded by brackets. Another section could contain merged billing information, not shown. The person who replies to the e-mail (the customer) is meant to include the corrections or additions within the indicated brackets.

By allowing the recipient to take advantage of the offer simply by replying to the e-mail, rather than requiring the recipient to place an order by linking to a related web-site or to print the e-mail message and FAX it back, or to call an 800 number, a much higher return rate can be achieved. For conventional outbound e-mail messages that require the recipient to click on an embedded URL to go to a web site, the returns may be on the order of several hundred percent on investment (the fee charged for delivering the outbound messages). By enabling the recipient to provide direct response e-mail messages as in figure 1, the return on investment can be as high as several thousand percent.

Figures 2A and 2B illustrate a similar outbound e-mail message in which there is no choice of products but only a single offer to be accepted or rejected. To take advantage of the offer, the recipient types "yes" in the subject line. In figure 2B, a shipping block 18 of the kind mentioned above is shown. (In this case, the shipping block contains no information because the shipping address is the same as the billing address.)

One reason for including differential billing and shipping blocks is to acquire information in the return e-mail message that is similar to information captured in orders placed on a related web site. In a system in which web-site orders generate fields that can be fed directly to an automated order fulfillment process, it is useful to make the e-mail message information field-wise consistent to permit the information to be delivered automatically to the same order fulfillment process.

Exception processing

Processing the inbound e-mails (the ones with responses concerning commercial transactions from the recipients of the outbound e-mails) may require custom interaction with the recipients. For example, the wording of the outbound messages may be confusing to the recipients.

As shown in figure 3, the system 40 enables the transactional e-mail message processor 42 to determine when a dialog with the recipient 44 is needed and then assists a human service representative 46 to conduct an effective dialog 48. The dialog can be conducted on behalf of the vendor 50 but without involving the vendor. Alternatively, the vendor's fulfillment process 52 can be notified electronically 54 of interaction that may be required. Easing the processing of responses that include customer orders is important because the orders typically come back quickly, e.g., within 36-48 hours, and in large volume. The ability to deal with questions that arise as a result of the contact from a customer service point of view keeps the vendor's customer service organization from being overwhelmed by the responses that come back.

The ability to process exceptions without involving the customer service organization of the vendor is based partly on knowing how the outbound e-mail messages were constructed. As a simple example, a recipient may ask an unnecessary question that could have been answered by reading the outbound e-mail message. The e-mail message processor can pull out the relevant portion of the message and send it back to the recipient to answer the question.

ProcOrder Process

The inbound e-mail messages 60 are batch processed by a script called ProcOrder 62. ProcOrder parses the elements of the inbound e-mail messages in accordance with the original set up and instructions of the outbound e-mail messages 64. ProcOrder determines if all of the items that are required for an order to be completely processed automatically appear in the inbound e-mail message. For example, the script would look for the ordering token, such as the word "yes" or a series of letters depending on whether it is a single or multiple offer. The script would also look for footer information in the e-mail message, including a code that identifies the given campaign and the given offer, as seen in block 66 of figure 2B. In that example, there are four components in the footer, but only two are represented because the other two are not required in this instance. The first element is a customer identifier 68, e.g. 861270. Then there is a space 70 between two pipes that would contain the list identifier if there were one. There may be multiple recipient lists for a given marketing campaign. In the example, there is only one list, and there is no list identifier. A list number 243 might refer to a list of people who made a purchase at the vendor's web site or who subscribed at the web-site for a listserv.

The third footer item could be a source of awareness code 72, e.g., 3275, which identifies a particular marketing campaign. For example, in the case of figure 2, the code could refer to a Benchmarking Three-part Video Series offer.

The last item in the footer, located between the final pipe and the first right bracket would be a flight identification code 74. A given campaign could have multiple flights of e-mail messages.

After looking for the footer information, the ProcOrder parser looks for fields in the billing and shipping address blocks that are required to complete the order. What is required may vary with the type of campaign but typically the minimum requirements are a name and a physical address. If the information is not completely available in the response e-mail message, the script checks to see if it is available in the database 76. If not available in either place, the script generates an exception entry for an exception list. The exception list is provided to a service representative 46 who can then act on it (without involving the vendor's customer service organization), e.g., by sending back an e-mail message asking for the shipping address.

If all required information is available, the script generates a fully fielded valid order in a format required by the fulfillment system of the vendor and adds it to a batch of valid orders 78 which are sent electronically to the fulfillment process.

<u>Confirmation e-mail message</u>

As a result of running the ProcOrder script, an e-mail message 80 is returned to each customer either to confirm an order or to request more information. In the latter case, a dialog ensues and is managed by software and through an exception handling service as explained earlier. For example, the customer's response could say something like "sure, send"; or "send it and I'll take a look." Shortly thereafter the customer would receive a confirmation "Thank you for your order; you can expect the CD-ROM in about seven business days. Please let us know if there is anything else we can do to help simply by replying to this e-mail."

One-click ordering

Another feature of the e-mail dialog with a customer involves simplifying and optimizing the presentation of content. In the examples of figures 1 and 2, the information is presented in a simple text format. It is useful also to provide in-line HTML code in the outbound e-mail message in a manner similar to the one-click ordering that Amazon.com offers in a web-site context. In one-click ordering, the customer sets up an account by providing credit card and shipping information. On subsequent visits to the web site, the customer can pick a product with one click, place an order, and have it shipped. A similar technique could be adapted to e-mail message interchange by embedding one-click ordering into e-mail.

An advantage of in-line HTML code is the opportunity for a much higher response rate because of the higher graphical contact and higher level of engagement normally achieved by a graphical message.

Template

The outbound e-mail messages are set up in a standard format using templates 90. The templates enable either a single-offer message or a multiple-offer message. Other templates are also possible, including one that embeds in-line HTML into the message as mentioned above, either for the single-offer or multiple-offer cases.

In addition, a set-up tool 92 permits the parameters of a given campaign to be defined, including the source of awareness code, the flight identification code, the campaign identification code, and similar information. The set-up tool also permits defining the tokens that are to be used in a given campaign (for example, the letters assigned to different products being offered). The set-up tool also allows a definition of the required fields that must appear in a given campaign to enable automated generation of orders to an existing fulfillment system.

The set-up tool also provides a user interface that enables a vendor to help in entering the set-up information.

The result of applying the tool to the templates is a set of outbound message forms 94 that are ready for use.

<u>Reporting tool</u>

After the template is set up and the system is ready to launch a flight, address 108 and other information 110. 112 stored in the target list of customers is merged with the message forms, and the e-mail messages are automatically generated and sent by an outbound e-mail delivery engine 96. Customers then begin to respond. The ProcOrder script generates automatic orders to the fulfillment system and exception information for additional processing.

A reporting tool 104 aggregates information about the responses for a given campaign according to source of awareness code and flight. The information is made available on-line to the vendor and can be used for a variety of marketing purposes. The information could be generated as an Excel file attached to an e-mail, or as a paper-based report, or as an electronic file that is transferred on a batch basis.

<u>Gathering additional information from database</u>

There may be an intermediate step between the parsing engine's (ProcOrder) extraction of information from an e-mail message and the generation of the valid order. The intermediate step could be a querying process 112 to gather additional information from an existing database. The additional information may not have been included in the outbound e-mail messages but may be needed to generate a valid order. For example, product codes 112 may be stored in the database but not included in the outbound e-mail message. The letters entered by the customer can be mapped to the actual product codes by reference to the tables of the database based upon the source of awareness code.

The resulting valid order is a fully-fielded record that has the fields required by the client's order fulfillment system to process an order.

<u>Exception treatment</u>

Exception handling can be treated in different ways depending on the circumstances. For example, an exception might occur when a customer responds from an e-mail client that does not quote the original text of the outbound e-mail message. The inbound e-mail message then has the customer's e-mail address, a subject line that says "yes", and the original subject line from the campaign, but does not have the required information for the shipping address or the footer information. ProcOrder would kick that out as an exception, but the exception handling system would allow a response management representative 46, based on the e-mail address, to confirm, from the database 76, that all of the required information is available. Use of the subject line allows the system to tie back to the appropriate campaign and to figure out who is ordering and what he is ordering. A valid order can be created without further interaction with the customer other than to send him a confirmation that the system has been able to enter a valid order on his behalf.

The system thus recognizes that it is not likely to be possible to automate every interaction with the customer, but it may be possible to complete a dialog with essentially all of the customers from whom inbound e-mail messages are received by automatically identifying messages that will require custom human handling and providing information and tools that enable the human handlers to complete the exception transactions in an efficient manner.

Non-order response processing

Not every inbound e-mail message is an order. Non-order messages include undeliverable bounced messages to ad hoc customer service responses. Non-order inbound e-mail messages must be identified by the parsing engine.

Undeliverable e-mail messages 114 are automatically separated from the inbound e-mail stream and stored for offline handling by a human response handling professional, who operates a script on the files of undeliverable messages. The script classifies them as "soft" and "hard," parses e-mail addresses and footer data from the messages, matches the parsed records to the database, and flags appropriate records as "undeliverable".

Other non-order messages also are handled manually as explained earlier.

Vendor creation of e-mail campaigns.

A campaign creation tool 126 is provided to a vendor to enable simple entry of all information needed to create an e-mail campaign, including all the parameters, the text of the messages, and the tables of data needed in the database. The vendor delivers the campaign electronically to the transactional e-mail processor which then delivers the e-mail messages, receive the responses, processes all exceptions, and returns to the fulfillment system the vendor orders in a proper format.

A web-based vendor interface 128 enables on-line viewing by the vendor of the status of all campaigns, including the state of those that are in development and the results of those that are "live". The information is hosted by the transactional e-mail processor in part based on the database 76. The interface also gives the vendor a mechanism to check text and other content into the database.

Alternatively, instead of automatically permitting the vendor to fully create a finished campaign, the vendor may be enabled to download and check into the database a proposed campaign. Then an account executive of the e-mail handler process would review it and work with the vendor to complete it before it is finally queued for distribution.

Appendices A, B, and C contain more detailed descriptions of aspects of implementations of the invention. Appendix D contains source code written of an example of the ProcOrder process.

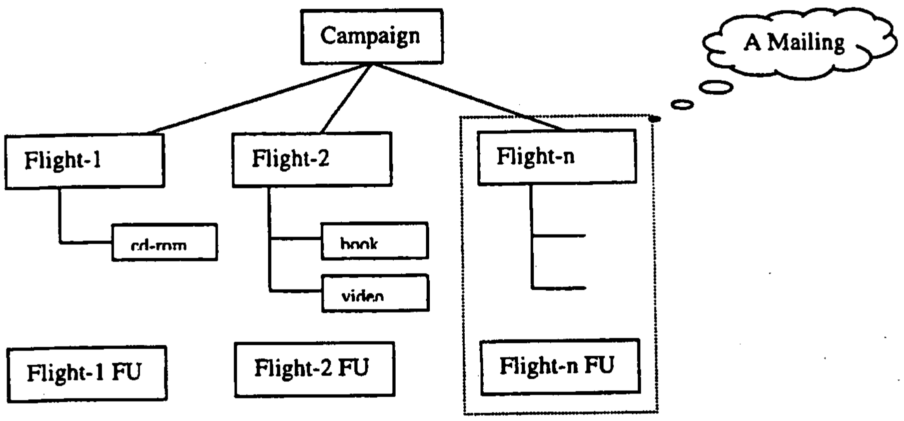
Other implementations are within the scope of the following claims.

What is claimed is:

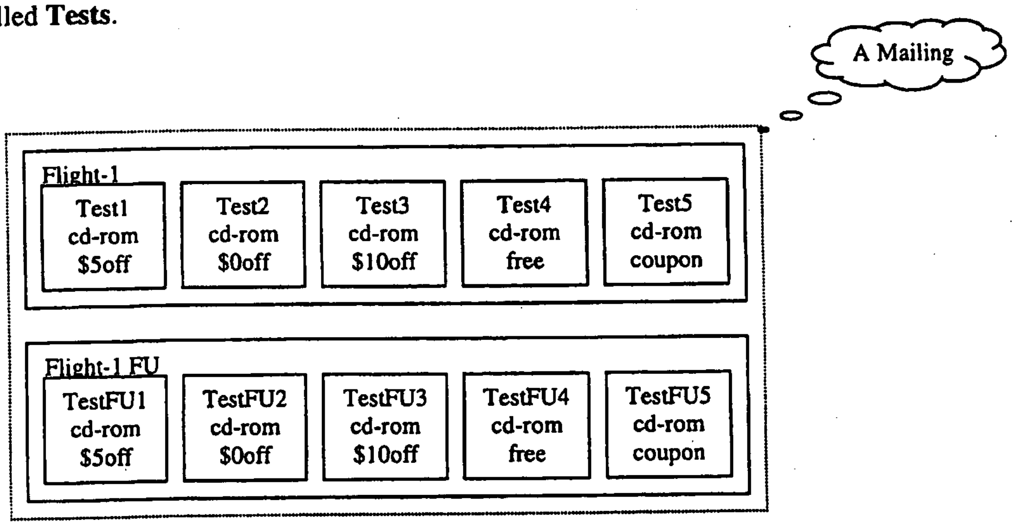
e-mail Protocols, Structures, Definitions & Cycles

MAIL CAMPAIGN

A mail campaign is defined in terms of **Flights.**



A flight contains one or more items that are being promoted but is not restricted to the manner in which the promotion is performed. A flight could be setup to promote a CD-ROM but has variances in the promotion of the item to the targeted mailing. These variances in a flight are called **Tests.**



In order to reference the entities above, each are assigned Ids as follows: APPENDIX A

A Campaign
- Has a Campaign ID , **CID**
- Has 1 to n flights and flight follow-ups

A Mailing
- Has a Mailing ID, **MID**
- Has a Client Code (SOAC). The mailing is performed on behalf of the client.
- Has a Flight ID, **FID** and a Flight Follow-up ID, **FID**

A Flight
- Has a flight ID, **FID**
- Shares a **SOAC** with its Flight Follow-up
- Contains one or more Test Ids, **TID**
- Has 1 to m items for sale

An Item
- Has an Item ID **IID**
- Has an Item code (determined by the client)

To summarize, the following ID acronyms are used:

-	MID	-	Member ID (ID of the person receiving the e-mail)
-	CID	-	Campaign ID
-	LID	-	maiLing ID
-	FID	-	Flight ID
-	TID	-	Test ID
-	IID	-	Item ID
-	SID	-	Style ID (ID of the style of the e-mail message)

MAIL PROTOCOLS

SMTP

The *Simple Mail Transfer Protocol* (SMTP) is described in RFC 821 and is the way that two sites on the Internet exchange mail messages.

The commands are:

- HELO	*domain*
- MAIL	FROM: *username*
- RCPT	TO: *username*
- DATA	
- QUIT	

Each command is terminated with a CR-LF pair. Replies start with a three-digit response code and continue with text designed to be read by users.

POP3

POP3 is the Post Office Protocol. If the site is always on the Internet, then mail would be sent with an SMTP-sender and received with an SMTP-receiver. However, it may cases it is not possible to maintain a permanent internet connection and in such cases, the Post Office Protocol is used to receive the inbound mail. POP3 allows mail to be stored on machine that is always on the Internet and a receiving host connects to it, asks for any mail and disconnects.

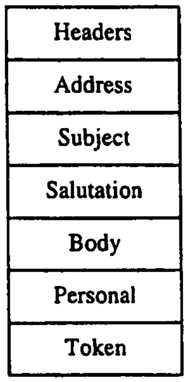
The commands are:

- USER	*username*
- PASS	*password*
- STAT	
- LIST	*[message number]*
- RETR	*message number*
- DEL	*message number*
- LAST	
- QUIT	

THE ANATOMY OF AN OUTBOUND E-MAIL MESSAGE

An outbound email message has a predefined structure. The message is created by combining together a text block with an email address.

The structure of an outbound mail message consists of a Property, Address, Subject, Header, Body, Personal and Token.



Structure of an outbound e-mail message

a) Headers

The first few lines of an Outbound Mail message are called the Headers and have a defined format. This information is not normally displayed to the User by a Mail Client application and can only be viewed if the Client permits e.g., in Microsoft Exchange this information can be viewed by listing the properties of a mail message as follows:

```
Received: by mail.kersur.net (mbox peterk)
 (with Cubic Circle's cucipop (v1.31 1998/05/13) Fri Mar 19 20:59:00 1999)
X-From_: aestes@e-dialog.com  Fri Mar 19 13:43:54 1999
Return-Path: <aestes@e-dialog.com>
Received: from montana.e-dialog.com (mail.e-dialog.com [207.31.244.2])
        by mail.kersur.net (8.9.1/8.9.1) with ESMTP id NAA10659
        for <peterk@sytech.com>; Fri, 19 Mar 1999 13:43:53 -0500 (EST)
Received: by MONTANA with Internet Mail Service (5.5.2448.0)
        id <GZZPKKZ2>; Fri, 19 Mar 1999 13:43:57 -0500
Message-ID: <B15DC0490C8AD211BDFD004005A0C2CC47F10B@MONTANA>
From: Anthony  Estes <aestes@e-dialog.com>
To: "'peterk@sytech.com'" <peterk@sytech.com>
Subject: FW: Warning: could not send message for past 4 hours
Date: Fri, 19 Mar 1999 13:43:54 -0500
MIME-Version: 1.0
X-Mailer: Internet Mail Service (5.5.2448.0)
Content-Type: multipart/mixed;
        boundary="----_=_NextPart_000_01BE7238.71C0EB9A"

This message is in MIME format. Since your mail reader does not understand
this format, some or all of this message may not be legible.

------_=_NextPart_000_01BE7238.71C0EB9A
Content-Type: text/plain;
        charset="iso-8859-1"

------_=_NextPart_000_01BE7238.71C0EB9A
Content-Type: application/octet-stream;
        name="ATT00547.TXT"
Content-Disposition: attachment;
        filename="ATT00547.TXT"
```

```
------_=_NextPart_000_01BE7238.71C0EB9A
Content-Type: message/rfc822

Message-ID: <B15DC0490C8AD211BDFD004005A0C2CC47EFF2@MONTANA>
From: Anthony  Estes <aestes@e-dialog.com>
To: 'Hillary Gaeth (E-mail)' <HGaeth@engage.com>
Subject: * Thurs, Fri ?
Date: Tue, 9 Mar 1999 08:04:20 -0500
MIME-Version: 1.0
X-Mailer: Internet Mail Service (5.5.2448.0)
Content-Type: text/plain

------_=_NextPart_000_01BE7238.71C0EB9A--
```

A header starts with a *field name* followed by a colon and the field body. The contents of the field body may be rigidly defined or free form.

The following headers are mandatory; that is, there must be a header with each of these names:
- Date
- From, or Sender and From
- To, or CC (carbon copy), BCC (blind carbon copy)

The following headers are optional:
- Return-path
- Received
- Reply-To
- Message-ID
- In-Reply-To
- References
- Keywords
- Subject
- Comments
- Encrypted

Some of these headers are obscure and rarely used. In addition, some mail clients generate their own extra headers. Many such extra header start with the characters "X-" because if extra mail headers are added to the RFC they will never start with these characters.

A header may be split over two lines according to the following rules:
- The split must be at a place where whitespace(blank or tabs) would normally occur; for example, not in the middle of a username or similar field.
- The continuation line must start with a space or tab.

Similary, a header can have extra white spaces almost everywhere except in the middle of a field.

Two headers that are given special attention in the context of this document are those that contain the address and subject.

b) Address
The address is a header that contains the email address of the person receiving the message.

c) Subject

This is a header that describes the subject of the mail. Contained in the subject is a **Subject Token** in the form of two characters. For example, for the mail subject '** A Special Offer for Selected Managers', the **Subject Token** is **

The **Subject Token** is chosen to identify the **Test ID** of the mail message.

d) Salutation

The salutation is a text block in the outbound mail message that is personalized to the person receiving the message. The intention is to provide a personalized greeting and an indication of the sender.

```
From: Harvard Business School Publishing <hbsp@e-care.com>
To: 'esalas@protelsa.com.pe'
Subject: ** A Special Offer for Selected Manager
Date: Tuesday, July 14, 1998 6:47 PM

From the Desk of Laura Winig
Harvard Business School Publishing Corporation
Boston, Massachusetts

Tuesday, July 14, 1998

Ms. Elizabeth Salas
5700 Collins Avenue  Apt  6h
Miami Beach, FL  33140-2308

Dear Ms. Salas:
```

The information in italic is personalized and is combined with the remainder of the message at the time of dispatch.

e) Body

The body contains the main core of the message. The format and layout is fixed and not personalized for a particular test.

```
Simply type "YES" in your reply to this e-mail to take "Virtual Work: Real Results" for a No-
Obligation Test Drive!
=-------------------------------------------------------------=
Do you work with colleagues and clients in multiple locations? Communicate more by email and
conference calls than through meetings? Find your "office" is wherever you are at any given
moment? Then you're working "virtually" and you know that working effectively without proximity
is essential in today's workplace. And you know it's not as easy as it looks.

Here at the publishing arm of Harvard Business School, we've combined extensive research and
real-life examples to create "Virtual Work: Real Results"-a dynamic multimedia program that can
improve your effectiveness working "virtually."
=-------------------------------------------------------------=
Use this engaging tool and you'll understand:

* the dynamics and politics of working virtually-
  and how to handle tough situations when you can't be face-to-face;

* how to most effectively use email, video conferencing,
  voice conferencing-and how to overcome fear of technology;

* ways to build relationships and trust without "human touch"-
```

```
 and how a virtual team can work efficiently and seamlessly.
=--------------------------------------------------------------=
We bring you tricks of the trade from the experts in working virtually.
Then, you gain confidence using these techniques through an interactive case study where you lead
a virtual team through a project. You'll make decisions that determine the success of its
efforts-all in a realistic, but no-risk, environment.

Overcome the isolation and conflicting loyalties that are inherent in working in a virtual
environment-and get ready for success with "Virtual Work: Real Results."
=--------------------------------------------------------------=
Take "Virtual Work: Real Results" for a No-Obligation Test Drive.
=--------------------------------------------------------------=
Simply reply to this email and we'll send you the program with our compliments. We're confident
you'll find you're working more effectively in the virtual world. After 14 days, we'll send you
an invoice for just $295 (single user license).

 But remember, if you're not entirely pleased with the program, simply call us and we'll arrange
                        to pick it up. You will owe nothing.

Sincerely,

Laura Winig
Director
```

In the above example the response with a "YES" is sufficient to indicate the purchase of the single product on offer. In a multi-product offer, the items would be listed and associated with a letter. For this type of mailing, the responder would list the letters in the response.

A mailing may have a follow up 'reminder' flight. This reminder would not go to respondents of the original flight. For example:

```
On Wednesday, July 1, I sent you a special offer on "Virtual Work: Real Results": a new
interactive CD-ROM from Harvard Business School Publishing.

Since I haven't heard back from you, I wanted to send you a reminder Before the offer expires.

If you are simply not interested, I apologize for the intrusion.

=---------    BELOW IS A REPRINT OF THIS SPECIAL OFFER    ---------=
Simply type "YES" in your reply to this e-mail to take "Virtual Work: Real Results" for a No-
Obligation Test Drive!
=--------------------------------------------------------------=
Do you work with colleagues and clients in multiple locations? Communicate more by email and
conference calls than through meetings? Find your "office" is wherever you are at any given
moment? Then you're working "virtually" and you know that working
...
...
...
confident you'll find you're working more effectively in the virtual world. After 14 days, we'll
send you an invoice for just $295 (single user license).

 But remember, if you're not entirely pleased with the program, simply call us and we'll arrange
                        to pick it up. You will owe nothing.

Sincerely,

Laura Winig
Director
```

f) Personal

The personal is added to the outbound message, after the mail body. The purpose of the text block is to request personal details from the respondent. The information requested is presented in two columns, the first indicating the type of information required and the second as a place for the reply to be entered (in [..]). For example,

```
FIRST NAME:          []
LAST NAME:           []
TITLE:               []
COMPANY:             []
DEPARTMENT:          []
ADDRESS1:            []
ADDRESS2:            []
ADDRESS3:            []
CITY:                []
PROVINCE/STATE:      []
POSTAL/ZIP CODE      []
COUNTRY:             []
PHONE:               []
FAX:                 []
EMAIL:               []
```

These details are to determine the shipping and billing information.

e) Token

The token follows the Mail Body (or Mail Personal if applicable) and contains information about the mailing and also the addressee e.g., `[[878119|2815|1]]`

The format of a Mail Token is

[[MID | TID | SID]]

where

- MID is an membership ID assigned to the person receiving the outbound message
- TID is the test ID assigned to the outbound message. This would be related to the Subject Token.
- SID is the style ID assigned to the style of the mailing i.e., single product or multi-product.

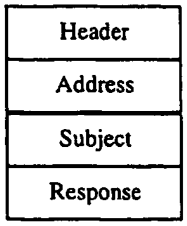
These three pieces of information uniquely identify the receiver of the outbound message and also the information they received. Hence, the processing of a response is greatly simplified if the reply returns the mail token.

In summary, an outbound message contains generic information that is the same in all the mailings of a test, and also personalized:

Generic	*Personalized*
Subject	Header
Body	Address
	Salutation
	Personal
	Token

THE ANATOMY OF AN INBOUND E-MAIL MESSAGE

An inbound email message has a predefined structure. However, the structure may not be 'structured' sufficiently for it to be automatically processed.



a) Headers
The first few lines of an Inbound Mail message are Headers and have a defined format. This information is similar in format to that of the outbound message. If the response is produced by replying to an outbound message, (instead of creating it from scratch) then it is probable that the headers of the outbound will be in the inbound mail.

b) Address
The address is a header that contains the email address of the respondent.

c) Subject
This is a header that describes the subject of the mail. This is free-form text and no assumptions can be on its structure. It may be the subject that was used in the outbound mail.

d) Response
The response is a text block containing the message from the respondent. It should not be assumed that the response has any structure since a responder has the freedom to write a reply in "free format" and is not forced to a guideline. This creates a number of problems for the processing of an inbound response since rigid rules cannot be applied.

The points that can be noted about a response are that:

- All responses will contain headers.

- All responses will contain the responders email address.

- All responses will contain a subject. The textual content of the subject cannot be assumed since it can be freely edited and so may not resemble the content of the outbound. If a subject contains a Subject Token then it should correspond to the TID.

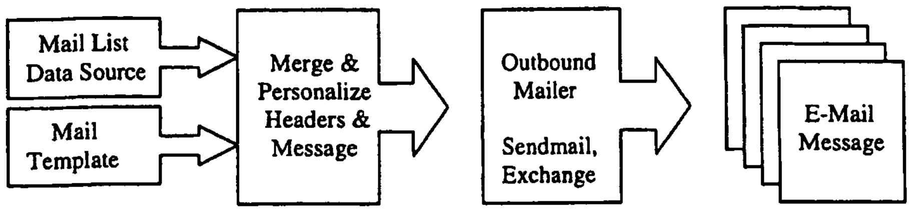
- If the response contains the Mail Token then the MID, TID and SID will be available and consequently it will be clear on the approach that should be taken in processing the response i.e., the handling of a single product 'YES' versus a multi-product 'ADG'.

AN E-MAIL LIFE CYCLE

OUTBOUND

a) Create an outbound message
An outbound email message consists of texts block that contains information and one or many questions or choices. The text block of an outbound mail message is sometimes referred to as a **Mail Template.**

b) Deploy the outbound message
The message can be deployed to anyone for whom an e-mail address is available. The address information in a **Mail List** (e.g., database, listserve) is merged with the **Mail Template** and sent. In some cases, the X-Headers are also personalized to reflect the purpose of the outbound mail message.



INBOUND

a) Receive a Reply
The content of a reply can have various forms

- The reply responds to the outbound message e.g., an order.
- The reply requests to unsubscribe from future mailings. The reply contains an indication that the respondent does not wish to receive any further mailings e.g., UNSUB, UNSUBSCRIBE, UNJOIN and REMOVE.
- The reply requests customer service and ad hoc questions.
- The reply indicates a change in personal details or to be added to the mailing list.

There is also the possibility that the outbound message did not reach its final destination and that it bounced back. There are two categories of bounces, hard bounces and soft bounces.

- A hard bounce notification indicates outright failure. For a hard bounce, the subject would contain a message of the following form:

```
Returned mail: Host unknown (Name server: ssoofftteecchh.com: host not found)
```

The sender of a hard bounce is usually specific such as:

```
Mail Delivery System [MAILER-DAEMON]
```

A hard bounce can also be detected in the response X-Header:

```
Received: by mail.kersur.net (mbox peterk)
 (with Cubic Circle's cucipop (v1.31 1998/05/13) Fri Mar 19 21:04:39 1999)
X-From_: MAILER-DAEMON  Fri Mar 19 21:04:36 1999
Return-Path: <MAILER-DAEMON>
Received: from localhost (localhost)
   by mail.kersur.net (8.9.1/8.9.1) with internal id VAA20320;
   Fri, 19 Mar 1999 21:04:36 -0500 (EST)
Date: Fri, 19 Mar 1999 21:04:36 -0500 (EST)
From: Mail Delivery Subsystem <MAILER-DAEMON>
Message-Id: <199903210204.VAA20320@mail.kersur.net>
To: <peterk@sytech.com>
MIME-Version: 1.0
Content-Type: multipart/report; report-type=delivery-status;
   boundary="VAA20320.921981876/mail.kersur.net"
Subject: Returned mail: Host unknown (Name server: ssoofftteecchh.com: host not found)
Auto-Submitted: auto-generated (failure)
```

The response body can also contain failure information:

```
The original message was received at Fri, 19 Mar 1999 21:04:35 -0500 (EST)
from dialup111.kersur.net [207.180.95.76]

   ----- The following addresses had permanent fatal errors -----
<JohnSmith@SSOOFFTTEECCHH.com>

   ----- Transcript of session follows -----
550 <JohnSmith@SSOOFFTTEECCHH.com>... Host unknown (Name server: ssoofftteecchh.com: host not
found)
```

- Softbounces can be of three types:
  - *NonDeliveryNotification* occurs when a given message has not been delivered yet but will continue to try and deliver for a further specified period of time. This state can be detected in the response subject:

    ```
    FW: Warning: could not send message for past 4 hours
    ```

    The response can also be detected in the response body:

    ```
    **********************************************
    **       THIS IS A WARNING MESSAGE ONLY      **
    **  YOU DO NOT NEED TO RESEND YOUR MESSAGE   **
    **********************************************
    The original message was received at Tue, 9 Mar 1999 08:00:18 -0500 (EST)
    from mail.e-dialog.com [207.31.244.2]

       ----- The following addresses had transient non-fatal errors -----
    hgaeth@andexc01.cmgi.com
        (expanded from: <HGaeth@engage.com>)

       ----- Transcript of session follows -----
    hgaeth@andexc01.cmgi.com... Deferred: Connection refused by
    andexc01.cmgi.com.
    Warning: message still undelivered after 4 hours
    Will keep trying until message is 3 days old
    ```

  - *AutoResponders* are notifications which actually indicate delivery but are sent by mail agents to indicate that the user will not be able to respond immediately (possibly on vacation) but the sender should expect a response when they return.
  - *Unknown*

b) Process the reply
Automatic processing of an e-mail response is defined as the ability to determine accurately the requirements of the reply by using text inspection and search rules.

To automatically process an e-mail response, there are a number of criteria that need to be satisfied by the content of the reply. The main three criteria are:

ci) Who is the Respondent
cii) What outbound message is the response to
ciii) What does the Respondent require

The exception to the above are hard and soft bounces that are identified by other e-mail properties.

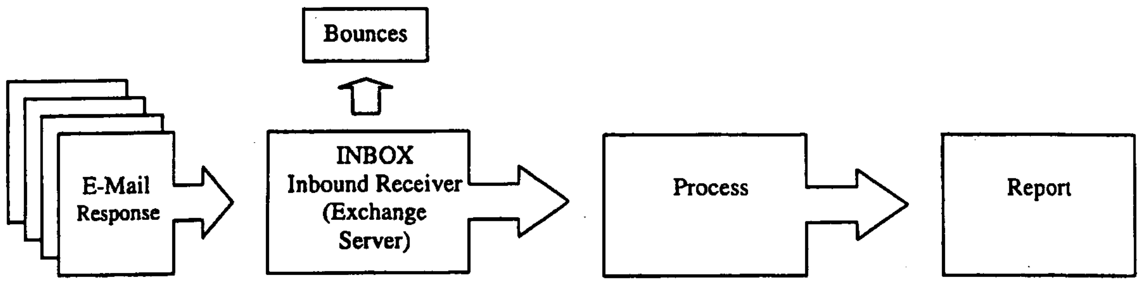
Since all legitimate mail responses will contain the e-mail address of the Sender this can be regarded as a base information for all responses. This information fulfills criteria ci) but alone is not sufficient for a response to be processed automatically.

Criteria cii) can be satisfied by the Subject Token, provided the respondent has not altered it. Note that the Style ID is implied if criteria cii) is satisfied.

Another source of information is the Mail Token. For a reply that contains a Mail Token, both criteria ci) and cii) can be determined.

But how is criteria ciii) satisfied? There is no simple solution to this question since the answer lies in the body of the reply, which is in 'free form'. The only method available that can determine the requirements of the respondent is to parse the email reply based on the SID.

c) Produce Reports
After determining the requirement of the inbound e-mail, database entries and reports can be produced.



Design Specification for QuickReply

QUICKREPLY

The QuickReply application is used to process e-mail responses to mailing campaigns. The building and sending of the e-mail messages is not within the scope of QuickReply.

The test within a campaign's flight provides the outbound e-mail message. The message will contain both generic and personalized information and will normally require a response from the person the message is sent to.

Responses to a test are received in the Inbox folder of the mailbox. The design objective of QuickReply is to read the responses from the Inbox, and decide from its content the requirements of the response. On determining (or not determining) the requirement, the response is filed to pre-configured sub-folders and the appropriate tracking database is up-dated.

QuickReply will generate diagnostic information that will support the reasoning on how it reached its decision.

The main data storage area of QuickReply is Microsoft Access relational database. Appropriate records will be added to this database as the response processing is performed. In addition, it is likely there will be responses that cannot be automatically processed and, in such cases, Customer Service will use QuickReply in manual mode. Manual mode will also add records to the appropriate tables in the database.

The information in the database will be used to generate reports.

APPENDIX B

MAILBOX ORGANIZATION

Mailings are performed on behalf of a Client. Consequently, all e-mail transfers for a particular Client will be performed within the Clients mailbox.

The organization of the Clients mailbox includes an Inbox for the responses and a pre-configured number of sub-folders where the responses are filed.

```
Mailbox - <Client Reference e.g., HBSP)
        Campaign CID
                Flight FID
                        AC
                                YYMMDDhh
                        AddChange
                                YYMMDDhh
                        BouncesHard
                                YYMMDDhh
                        BouncesSoft
                                YYMMDDhh
                        Master
                        Order
                                YYMMDDhh
                        Unclear
                                YYMMDDhh
                        Unsubscribe
                                YYMMDDhh
        Inbox
                YYMMDDhh
```

In the above, YYMMDDhh is a directory named after the year, month, day and hour. This format reflects the date and time the entries were placed in that folder.

Each test flight will have a number of responses associated with it. When QuickReply categorizes a response, it will be automatically moved from the Inbox to the appropriate sub-folder.

- **AC**	Responses for customer service to process manually
- **AddChange**	Responses that contain personal detail changes
- **BouncesHard**	Hard bounces
- **BouncesSoft**	Soft bounces
- **Master**	Master version of the outbound mail
- **Order**	Responses with an order
- **Unclear**	Responses that are unclear as to how they should be processed
- **Unsubscribe**	Response to unsubscribe

E-MAIL STYLES

The Style ID (SID) of a response indicates the approach that should be taken by QuickReply during processing.

The SID is a two-digit numerical value

XY

where:

- X=1 indicates the outbound Personal block is included in the outbound message.
- Y=message layout style described below.

The following two layout styles are supported:

a) Single Product Offer – layout style=1
This style offers one product. The respondent is asked to reply "YES" if they are interested.

b) Multi-Product Offer – layout style=2
The mailing information is based on information in a relational database. This style offers a number of products the respondent can choose from. For example, the outbound message could contain:

```
Your choices (detailed below) are:

A:      The work of Leadership Audio
B:      Leading Your People Audio
C:      Leading Change Successfully Audio
D:      Overcoming Resistance to Change Audio
E:      Gaining Competitive Advantage Audio
```

The respondent makes a choice and responds by stating their requirement in the free format e.g., "please send me ABE".

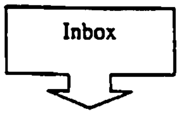
The number of items can vary as well as the choice symbol i.e., 1,2,3, could be used instead of A,B,C.

PROCESSING INBOUND MAIL AND REPORTING

The general processing and reporting function is described below. The general flow in processing responses is to determine the membership ID (MID), then the test ID (TID) and style ID (SID) and then perform an examination of the response to determine the requirement.

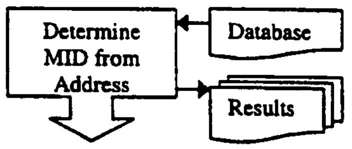
The processing criteria can be summarized as follows:
ci) who is the Respondent?
cii) what is the response to?
ciii) what does the Respondent require?

All inbound responses are targeted to the Inbox of the Client's Mailbox. On a periodic basis, the responses in the Inbox will be processed.

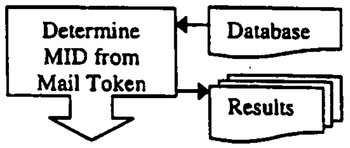


ci) Who is the respondent? – Determine the MID
The are several ways in which the Membership Information can be determined. The importance of the MID is that could provide a handle to the Members personal information

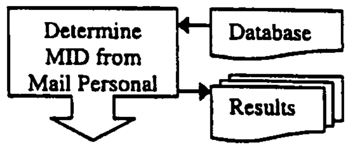
a) Extract the Response Address from the response. Determine if a unique record exists for the Address. If a unique address exists then the MID is determined. If the record is not unique then examine the mail token.



b) If the Mail Token exists then the MID is determined. Use the MID to confirm that the respondents e-mail address is the same as that in the database. If the addresses are different then report that this was the case but continue to process.



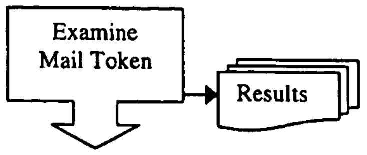
c) If the SID indicates that the Personal block was included in the outbound, the extract the Member details from the Mail Personal. The extraction assumes that the response information is in square brackets '[]'. The field definitions are defined in the database together with an indication of those that have to be completed by the responder. If Mail Personal is not complete (the fields are indicated in the database), move the response to Unclear and report.



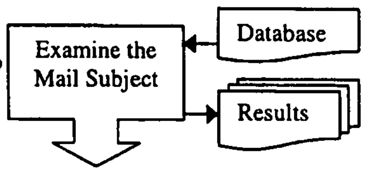
d) If the MID cannot be determined, move the response to Unclear and report.

cii) What is the response to? – Determine the TID and SID
The next stage in response processing is to determine what the response is to. This can be determined from the Test ID.

a) If the Mail Token exists then extract the TID and SID. If the Mail token is absent, report but continue the determination of the TID.



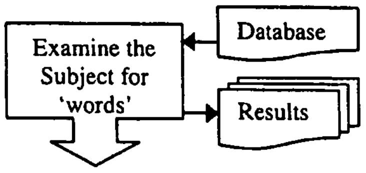
b) Determine the TID from the database based on the Mail Subject of the response. If the look-up is successful, then the SID will also be determined.



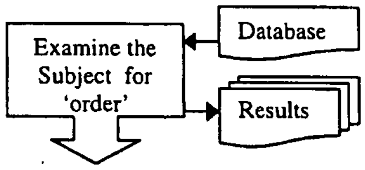
c) At this stage the TID and SID should be known. If not, move the response to Unclear and report.

ciii) What does the Respondent require?
With the MID, TID and SID determined, the response message(s) are still of no value until it is established what the respondent requires. The obvious approach to this problem is to compare the response with the original and try to make some sense of the differences.

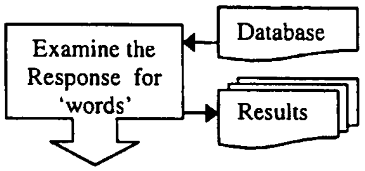
a) If the Mail Subject did not contain the Subject Token, examine for words that appear in the word action table. Any bounced responses are moved to the Bounce sub-folders. Any unsubscribe requests are moved to the Unsub sub-folder.



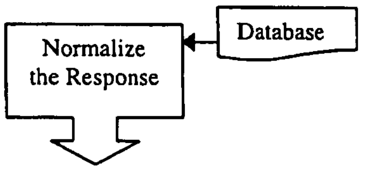
b) If the SID indicates a single item promotion, examine the Mail Subject for words that appear in the word action table. Any order responses are moved to the Orders sub-folders.



c) Examine the response for words that appear in the word action table. Any bounced responses are moved to the Bounce sub-folders. Any unsubscribe requests are moved to the Unsub sub-folder.

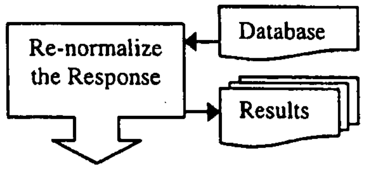


d) Remove all the text of the outbound message from the response.

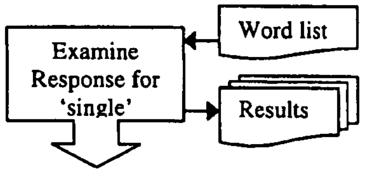


e) Normalize the response. Remove characters from the beginning of each line. The list of characters to be removed are in the word replacement table with ModeID=LineStart.

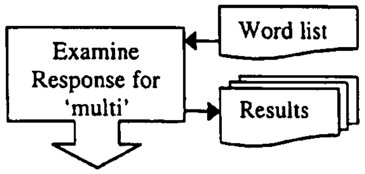
f) Replace phases from the normalized response. The phases that are replaced are given in the word replacement table with ModeID=PhaseReplace.



g) If the SID indicates that there is only a single item on promotion, examine the normalized response for words that indicate an order of a single product item. The words are in the word action table with the ActionID=SingleOrder. If order



h) If the SID indicates that there are multi-items on promotion, examine the normalized response for phrases that indicate an order of multi-items. The words are in the word action table with the ActionID=MultiOrder. In the case of multi-products, the



information the StartOption and EndOption from the bindings table are used to verify that item(s) ordered are in the correct choice range.

i) Any order requests are moved to the Orders sub-folder.

j) If the purpose of the response cannot be determined, then move the response to the Unclear sub-folder.

DATABASE ORGANIZATION

QuickReply uses two types of databases that are stored in Microsoft Access. The first database is part of the QuickReply application and contains high-level information on the campaigns, flights and tests. The second database is specific to a particular Client Test.

a) QuickReply Database

The QuickReply database contains the information that binds campaigns, flights and tests.

tblTestBindings	- **contain Test bindings**
Test ID	**unique**
Client ID	
Client Test Database	database name used for the test information
Subject	subject text of the outbound test
Style ID	the style of the outbound message
StartOption	character of the first item in the test
EndOption	character of the last item in the test
Campaign ID	
Mailing ID	
Flight ID	

tblOutboundPersonal	- **contains personal details that appear in a test**
Test ID	**unique**
Question	prompt displayed in personal block e.g., First Name
AnswerLen	the maximum length of the answer
AnswerReq?	flag to indicate that an answer must be given

tblWordActions	- **contains words that convey an action**
Word	**unique**
Action ID	action id for this word e.g., 'bounce'

tblWordReplacement	- **contains phrases and their replacement**
Phase	**unique**
Replacement	replacement to the phrase
Mode ID	type of replacement

b) Client Test Database

tblMembers	- **contains member details**
Member ID	**unique**
Email	
Prefix	
First	
Middle	

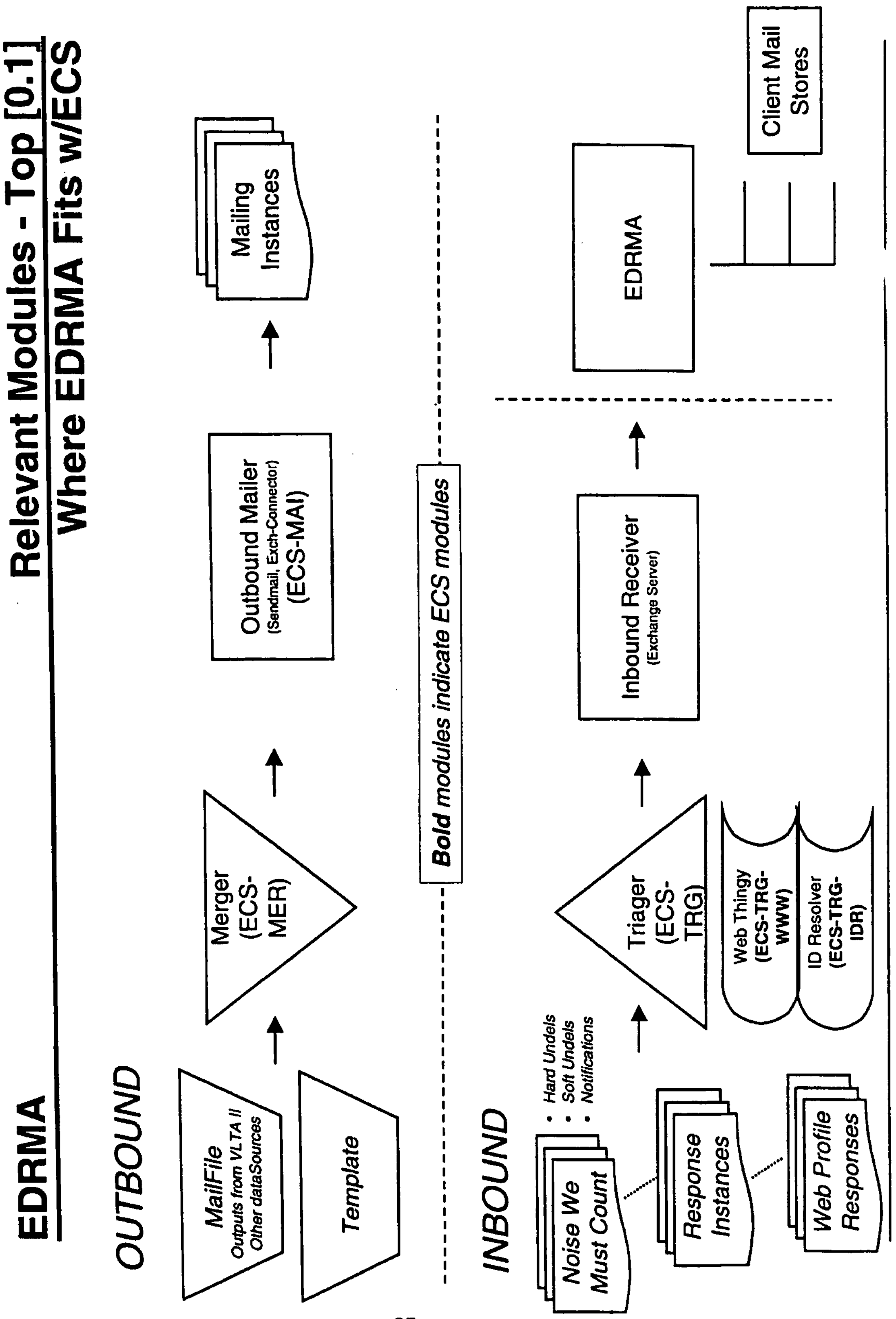
Last	
Suffix	
Title	
Company	
BillAddress1	ShipAddress1
BillAddress2	ShipAddress2
BillAddress3	ShipAddress3
BillCity	ShipCity
BillState	ShipState
BillZip	ShipZip
BillCountry	ShipCountry
Tel	
Ext	
Fax	

EDRMA — Introduction

This document is intended to provide a schematic synopsis of E-Dialog Response Management Architecture (EDRMA).

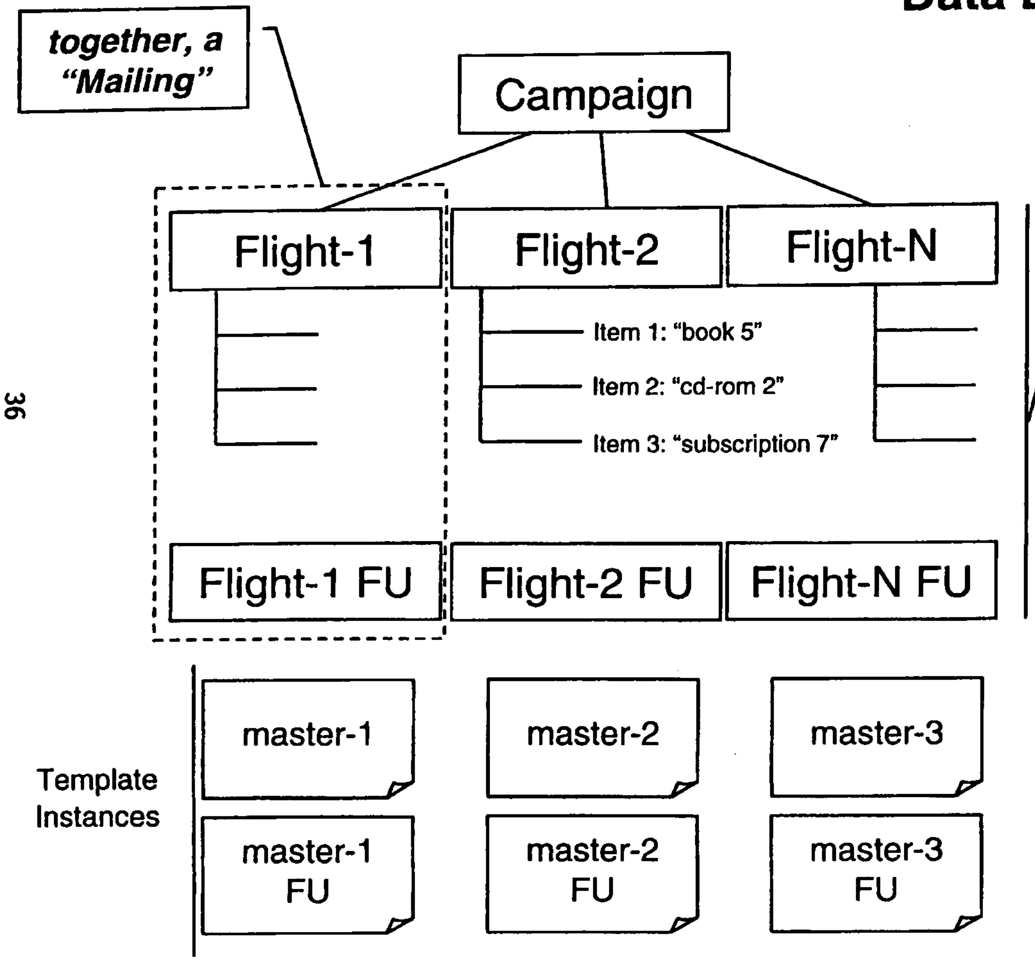
Thereby it is hoped that the design of the Verbind E-Mail Channel Server (ECS) can best accommodate EDRMA's input requirements, and so EDRMA's output requirements may be adjusted to match those of Verbind LifeTime's architecture (VLTA)

"First, some acronyms…" - *Anonymous*


EDRMA
Relevant Modules - Top [0.1]
Where EDRMA Fits w/ECS
OUTBOUND
MailFile
Outputs from VLTA II
Other dataSources
Template
Merger (ECS-MER)
Outbound Mailer
(Sendmail, Exch-Connector)
(ECS-MAI)
Mailing Instances
Bold modules indicate ECS modules
INBOUND
Noise We Must Count
Hard Undels
Soft Undels
Notifications
Response Instances
Web Profile Responses
Triager (ECS-TRG)
Web Thingy (ECS-TRG-WWW)
ID Resolver (ECS-TRG-IDR)
Inbound Receiver
(Exchange Server)
EDRMA
Client Mail Stores

EDRMA Model

Data Entity Relationships [0.5]



Campaign
- has 1 CID
- has 1:N Flights (FIDs)
- has 1 template class (TCID)
  - eNews
  - QuickReply - single
  - QuickReply - multi
- Approved Response Templates (template ID)

Mail Key Codes
- [[CustID|ListID|FID|TCID] <now>
- [[**MIID**|FID|TCID]] <future>
- **MIID** -> global mail instance ID
- placement: X-Header, Subject, Footer

Mailing (e.g., F1 & F1-FU)
- has 1 MID
- have 2 FIDs, one for F, FU
- has one SOAC (client code)

Any Flight has 1:N items for sale
- has one FID
- has one TCID
- (typ) shares a SOAC with its FU
- Any Item
  - has 1 InstanceID (our code)
  - has 1 ItemID (client code)

EDRMA **Mailbox**

Canonical Data Structures - Exchange Server Data Store [0.2]

Mailbox - <ClientFullName> *(not alias: "hbsp")*

<CampaignName>[<CID>]

- **ApprovedResponseTemplates <templateID>**
- **<FlightName>[<FID>]**
  - **AC** *(i.e., "Advocacy Care")*
    - + <date: YYMMDDhhmm>
  - **ADDCHANGE**
    - + <date: YYMMDDhhmm>
  - **HARDBOUNCES** *(i.e., hard undelivereds... "jsmith@foo.com is not a valid addressee")*
    - + <date: YYMMDDhhmm>
  - **INBOX (monolithic, this Mailbox)**
  - **MASTER (contains master template, this FID)**
  - **ORDERS**
    - + <date: YYMMDDhhmm>
      - » [[custIDlListIDlFlightIDlTCID]] << footer tags (x-header, subject)
  - **SOFTBOUNCES**
    - + **DELIVERYNOTIFICATIONS**
      - » <date: YYMMDDhhmm>
    - + **AUTORESPONDERS**
      - » <date: YYMMDDhhmm>
    - + **UNKNOWN**
      - » <date: YYMMDDhhmm>
  - **UNCLEAR**
    - + <date: YYMMDDhhmm>
  - **UNSUBS**
    - + <date: YYMMDDhhmm>

EDRMA Selected Applications

VBA.PKinboxInspector
- interface: GUI / VBA
- data: MAPI
- inbox sorting, folder management application

VBA.PKresponseProcessor
- interface: GUI / VBA
- data: MAPI
- response review and report preparation application

PL.AEprocOrder()
- interface: commandline
- data: ADO 2.0
- process raw "order e-mails" by TCID
- uses cf file rulesets for document preprocessing and data element parsing tied to CID; other rules, hardcoded
- generates:
  - Acceptable output for review, annotated
  - Exceptions, annotated
  - Truncated BODYs, annotated
  - Raw fields output , annotated
  - Raw fields exceptions , annotated
  - Rule-eval log (exhaustive)

PL.AEprocBatPrep()
- interface: commandline
- data: ADO 2.0
- takes selected output from PL.AEprocOrder() and transforms to a batch transfer specification via a field map and transform rules cf

PL.AEprocUpdateBatVerbInd()
- TBD

PL.AEscrubNormalCanon()
- << not used on the response side >>
- interface: commandline
- data: file handles
- scrubber, normalizer, canonicalizer
- also generates scrambled (non-predictable) row Ids and flags AOLs

EDRMA

Process Stages [0.5]

Preliminary ECS hooks anticipated

Preprocess Stage
- inspect Inbox using VBA.PKinboxInspector
  - auto-creates folder structure (previous slide)
  - facilitates auto-sort of items into ORDERS, UNSUBS, UNDELS, HARDS, SOFTS, ADDCHANGE, etc.
  - facilitates manu-sort of items into AC, and exception
- report preparation by TCID via AEprocOrder()
  - produces: OUT_report, EXC_report && EXC_BODY diagnostic report

Processing Stage using VBA.PKresponseProcessor
- for each EXC_
  - inspect, correct, reconcile (via Mailing Table lookups) and commit
- for each OUT_
  - inspect and commit to report (or not)
- Acceptable UNION of EXC_ && OUT_ -> Reporting, Update Stages { AEprocBatPrep() II AEprocUpdateBatVerbind() }
- Hard exceptions are tagged as such and become Followup RFC's to get additional (critical information)
- FUP (FUPID -> RFC) tags affect routing; tie-back to HARD exception row in this EXC_ report for closure

Response Follow-up Confirmation Stage
- for each non-BOUNCE (ORDERS, UNSUBS, etc.), respond to respondents with an appropriate "confirmation of receipt/action taken" message
- using relevant response template store (**templateID**)
- this class of response likewise tagged for routing (FUPID -> confirm)

Reporting Stage
- produce order report
  - deliver via fax
  - post to client private web application
  - deliver as formatted batch

Update Stage
- deliver formatted batch to {Verbind, Database} -> sp_???, SQL Executive

APPENDIX D

F:_Development\PRJ\PRJ_hbsp_extractOrder\WORKING\procOrder.pl
Printed at 19:48 on 06 Feb 1999

```
#
# procOrder.pl (c) 1998,1999 E-Dialog, Inc. All Rights Reserved.
#
# PURPOSE DISCUSSION
#
# program accepts a raw email stream from MS Exchange Server via export to ADO-compliant datastore
# then uses email body preprocessing and extraction rules - tailored per campaign in ./cf/ - to construct order reports.
#
#       cf files are campaign optimized:
#
#       mailingDBLayout.cf                    -> field mappings for a campaign's master mailing table
#       messageBodyDataLayout.cf              -> DSNs, field mappings, critical rule designations, order-designation bounds, synonymn table
#       messageBodyIndicateAddressChange.cf   -> rulesets for indicating ADDRESS CHANGES
#       messageBodyIndicateBuyAll.cf          -> rulesets for indicating BUY ALL state
#       messageBodyPreProcRules.cf            -> body text preprocessor configuration and rulesets
#
# program produces diagnostic and order-entry bound output files and logs outcomes of all ruleset evaluations leading to
# preparation of those output files, which include:
#
#       <$ARGV[1]>              -> summary order report file (designed for data entry, faxing, emailing to fulfillment)
#       EXC_<...>               -> (companion) summary exception report (same format as s. order report with annotations to indicate rule failures for man
ual review)
#       OUT_<...>               -> detailed order stream file (all fields discretely mapped and normalized for input into batch postprocessor)
#       EXC_DETAIL_<...>        -> (companion) ..exceptions..discrete fields, etc.
#       EXC_BODY_<...>          -> annotated diagnostic report containing all message bodies - sans original message - for aid in resolving exceptions man
ually
#
# output files are (default) tab-delimited text
#
# PROGRAM REQUIRES
#
#       Perl 5.003 interpreter later compiled for WIN32
#       MS ADO components 1.5 (2.0+ recommended)
#
# RELEASE HISTORY
#
# 981020AE: 0.5         first usage: to extract carol10 order reports
# 981204AE: 0.9         upgrade to handle carol12, multiple item selections
# 981208AE: 1.1         improve flagging
# 981211AE: 1.5         adapt to extract IDEAS@WORK add/changes
# 981222AE: 2.3.1       fix bugs in EXC_BODY reporting / enhance it
# 981222AE: 2.4         add cfPATH support
# 981223AE: 2.4.1       cfPath logging; fix synonymn (SUBJECT TOKEN) lookup <trailing \s*>
# 981223AE: 2.5         reorder reporting fields
# 981224AE: 2.5.1       reorder reporting fields
# 981225AE: 2.5.2       update SHIPPING_ADDRESS fields to reflect presence of T,B & S diagnostic fields
# 981226AE: 2.6         code review/dox; update cf to require MAILINGID
# 990106AE: 2.6.5       standardize on LOG ID for use in address change
# 990129ae: 2.7         add support for separate billing, shipping addresses
# 990201ae: 3.0         add record-recovery via ADO -> mailing database lookups
# 990202ae: 3.0.1       record-update ADO -> mailing database
# 990202ae: 3.0.2       fix leading digits bug in getDateTimeStamp ftn
# 990203ae: 3.1         fix MDB query bugs; introduce detection of SELECT && UPDATE SUCCESS; verify decision-tree based UPDATES
# 990205ae: 3.1.1       fix address3 bug; provide regression compatibility for /ADDCHANGE
# 990206ae: 3.1.2       throw .noTok as a HARD_EXC

#################### INCL #####################

use win32::OLE;

#################### ENV  #####################

$GrequiredARG   = 4;
$Grevision      = '3.1.2';

if(scalar(@ARGV) < $GrequiredARG) { die "($Grevision) Usage: procOrder.pl [userName] [reportName] [cfPATH <campaignMnemonic>] [ <buyToken> | /multi | /add
Change ]\n"; }

$Guser                          = $ARGV[0];
```

```
$GreportFile                    = $ARGV[1];
$GcfPath                        = $ARGV[2];
$Gsubj_buyToken                 = $ARGV[3];
$GexcReportFile                 = 'EXC_'.$GreportFile;
$GexcBODYReportFile             = 'EXC_BODY_'.$GreportFile;
$GoutFile                       = 'OUT_'.$GreportFile;
$GexceptionsFile                = 'EXC_DETAIL_'.$GreportFile;

if($ARGV[4] == 1) { $GDEBUG = 1; }      else {$GDEBUG = 0;}
if($ARGV[4] == 2) { $LGDEBUG_FTN = 1; } else {$LGDEBUG_FTN = 0; }

#################### DEFN ######################

## MISC GLOBALS ##

$Gplatform                      = 'WIN';
$GeventType                     = 'INFO';
$basePath                       = '.';
$GfieldDelimiter                = "\t";
$Gspace                         = ' ';
$Gnewline                       = "\n";
$Gcomma                         = ',';
$Gpipe                          = '|';
$eventRequiresFUP               = '';
$GmultiItemOrderSwitch          = '/MULTI';
$GaddChangeSwitch               = '/ADDCHANGE';

############## OUTPUT FILE LAYOUTS ###############

## DO NOT change alpha ordering of lowercase field names (e_ .. t_)
## This is vital in the BODY field layout defn file since the rules reference the letter prefixes

## reportFile (note: EXC_reportFileName also written from this template)

        %GreportFileRecord = (

                a_LID                      => '_e_$thisLogKey',
                ab_LINE                    => '<fill-in>',
                b_RID                      => '_e_$thisRecordID',
                c_THIS_SUBJECT             => '_e_$thisSubject',
                d_THIS_BODY_BOTTOM         => '_e_$thisBodyBottom',
                e_THIS_BODY_TOP            => '_e_$thisBodyTop',
                ia_FUP_REQUIRED            => '_e_$eventRequiresFUP',
                ib_LOCATION_BUY_TOKEN      => '_e_$thisBuyStatus',
                j_REVIEW_ACTION_TAKEN      => '<review>',
                za_SELECTED_ITEMS          => '_e_$thisItemsSelectedStack',
                zd_PRIORITY                => '_e_$thisMailingID',
                zf_CUSTNO                  => '_e_($thisCustNo==-1 || $thisCustNo == 0)?"":$thisCustNo',
                zi_LISTID                  => '_e_($thisListID==-1)?"":$thisListID',
                zm_FROM_ADDRESS            => '_e_$thisFromAddress',
                zp_SPECIAL_INSTRUCTIONS    => '<none>',
                zr_PHONE                   => '_e_$results{\'r_bphone\'}',
                zx_BILLING_ADDRESS         => '_e_$thisBillingAddressBlock',
                zy_SHIPPING_ADDRESS        => '_e_$thisShippingAddressBlock',
                zz_MDB_BILLING_ADDRESS     => '_e_$thisMDBbillingAddressBlock',
        );


## outFile

        %GoutFileRecord = (

                a_LID                      => '_e_$thisLogKey',
                aa_RID                     => '_e_$thisRecordID',
                ab_THIS_SUBJECT            => '_e_$thisSubject',
                ac_THIS_BODY_BOTTOM        => '_e_$thisBodyBottom',
                ad_THIS_BODY_TOP           => '_e_$thisBodyTop',
                b_LISTID                   => '_e_$thisListID',
                bb_CUSTNO                  => '_e_$thisCustNo',
                c_PRIORITY              => '_e_$thisMailingID',

                e_bfirst                => '_e_$results{\'e_bfirst\'}',
                f_bmiddle               => '_e_$results{\'f_bmiddle\'}',
                g_blast                 => '_e_$results{\'g_blast\'}',
                h_btitle                => '_e_$results{\'h_btitle\'}',
                i_bcompany              => '_e_$results{\'i_bcompany\'}',
                j_bdepartment           => '_e_$results{\'j_bdepartment\'}',
                k_baddress1             => '_e_$results{\'k_baddress1\'}',
                l_baddress2             => '_e_$results{\'l_baddress2\'}',
                m_baddress3             => '_e_$results{\'m_baddress3\'}',
                n_bcity                 => '_e_$results{\'n_bcity\'}',
                o_bstate                => '_e_$results{\'o_bstate\'}',
                p_bpostal               => '_e_$results{\'p_bpostal\'}',
                q_bcountry              => '_e_$results{\'q_bcountry\'}',
                r_bphone                => '_e_$results{\'r_bphone\'}',
                s_bfax                  => '_e_$results{\'s_bfax\'}',
                t_bemail                => '_e_$results{\'t_bemail\'}',

                ue_sfirst               => '_e_$results{\'ue_sfirst\'}',
                uf_smiddle              => '_e_$results{\'uf_smiddle\'}',
                ug_slast                => '_e_$results{\'ug_slast\'}',
                uh_stitle               => '_e_$results{\'uh_stitle\'}',
                ui_scompany             => '_e_$results{\'ui_scompany\'}',
                uj_sdepartment          => '_e_$results{\'uj_sdepartment\'}',
                uk_saddress1            => '_e_$results{\'uk_saddress1\'}',
                ul_saddress2            => '_e_$results{\'ul_saddress2\'}',
                um_saddress3            => '_e_$results{\'um_saddress3\'}',
                un_scity                => '_e_$results{\'un_scity\'}',
                uo_sstate               => '_e_$results{\'uo_sstate\'}',
                up_spostal              => '_e_$results{\'up_spostal\'}',
                uq_scountry             => '_e_$results{\'uq_scountry\'}',
                ur_sphone               => '_e_$results{\'ur_sphone\'}',
                us_sfax                 => '_e_$results{\'us_sfax\'}',
                ut_semail               => '_e_$results{\'ut_semail\'}',

                ux_FROM_ADDRESS         => '_e_$thisFromAddress',
                uy_LOCATION_BUY_TOKEN   => '_e_$thisBuyStatus',
                v_SUBJECT               => '_e_$thisSubject',
                z_FUP_REQUIRED          => '_e_$eventRequiresFUP',
                zz_REVIEW_TYPE          => '_e_$EXCEPTION_TYPE',
        );

## exceptionsFile - BODY only

        %GexceptionsBODYFileRecord = (

                a_LID                   => '_e_("==========================================================================================================
===========\n>>".$thisLogKey."<<")',
                aa_REVIEW_TYPE          => '_e_$EXCEPTION_TYPE',
                ab_EXCEPT_FLAGS         => '_e_$eventRequiresFUP',
                b_RID                   => '_e_("<<".$thisRecordID.">>")',
                u_FROM_ADDRESS          => '_e_("||".$thisFromAddress."||")',
                v_SUBJECT               => '_e_("||".$thisSubject."||\n========================================================================================
================================")',
                w_BODY_BELOW            => '_e_$thisFullNumberedBody',              # full body

        );

## activityLogFile

        %GactivityLogRecord = (

                a_LID                   => '_e_$thisLogKey',
                b_user                  => '_e_$Guser."/".$GcfPath',
                c_dateStamp             => '_e_&getDateStamp(\'WIN\',$LGDEBUG_FTN)',
                e_RID                   => '_e_$thisRecordID',
                f_clientName            => '_e_$GclientName',
                g_eventType             => '_e_$eventType',
```

F:_Development\PRJ\PRJ_hbsp_extractOrder-WORKING\procOrder.pl
Printed at 19:48 on 06 Feb 1999

```
                h_eventDesc             => '_e_$eventDesc',
                z_FUP_REQUIRED          => '_e_$eventRequiresFUP',
        );


#################### DATA ####################

&getKeyPos_Outfile(\%GoutFileRecord);           # identify the fieldwise position of fields in the outfile for canonicalization (use EXC file as model)

##print "--- debug --- FRM_ADD FIELDPOS= $fromAddressFieldPos\n";

$GDSN                           = 'dsnOrderProcessing';

if ($Gplatform ne 'WIN') {$relativePath = '/logs/'} else { $relativePath = '\\logs\\';}

$GactivityLogFileName           = $basePath.$relativePath.'activityLog.tab';
$GactivityLogKeysFileName       = $basePath.$relativePath.'activityLogKeys.tab';


if ($Gplatform ne 'WIN') {$relativePath = '/db/'} else { $relativePath = '\\db\\';}

$GexcReportDBkeysFileName       = $basePath.$relativePath.'excReportDBkeys.tab';
$GexceptionsDBkeysFileName      = $basePath.$relativePath.'excDetailDBkeys.tab';
$GexceptionsBODYDBkeysFileName  = $basePath.$relativePath.'excBodyDBkeys.tab';
$GoutFileDBkeysFileName         = $basePath.$relativePath.'outFileDBkeys.tab';
$GreportFileDBkeysFileName      = $basePath.$relativePath.'reportFileDBkeys.tab';

if ($Gplatform ne 'WIN') {$relativePath = 'cf'.'/'.$GcfPath} else { $relativePath = '\\cf\\'."$GcfPath\\";}

$GbodyDataLayoutFileName        = $basePath.$relativePath.'messageBodyDataLayout.cf';
$GbodyPreProcRulesFileName      = $basePath.$relativePath.'messageBodyPreProcRules.cf';
$GbodyBuyAllFileName            = $basePath.$relativePath.'messageBodyIndicateBuyAll.cf';
$GbodyAddressChangeFileName     = $basePath.$relativePath.'messageBodyIndicateAddressChange.cf';
$GmailingDBLayoutFileName       = $basePath.$relativePath.'mailingDBLayout.cf';

##################################################
#################### MAIN ####################
##################################################

$MODE                   = '';
$MULTI_BUY_ON           = 0;
$ADD_CHANGE_ON          = 0;
$Gsubj_buyToken         = uc($Gsubj_buyToken);

$MODE = "<TOKEN = |$Gsubj_buyToken|>";
if($Gsubj_buyToken =~ /\/MULTI/)        { $MULTI_BUY_ON = 1; $MODE .= '<MULTI BUY ON>';}
if($Gsubj_buyToken =~ /\/ADDCHANGE/)    { $ADD_CHANGE_ON = 1; $MODE .= '<ADDCHANGE ON>'; }

print "[procOrder R.$Grevision] $MODE\n".($GDEBUG?"\n<DEBUG: $GDEBUG:$LGDEBUG_FTN>\n\n":"\n");

## open CF files

print "** Opening: $GbodyPreProcRulesFileName...\n";
        @GPreProcRules                          = &loadFile_SCALAR($GbodyPreProcRulesFileName);

print "** Opening: $GbodyBuyAllFileName...\n";
        @GBuyAllIndicators                      = &loadFile_SCALAR($GbodyBuyAllFileName);

print "** Opening: $GbodyAddressChangeFileName...\n";
        @GAddressChangeIndicators               = &loadFile_SCALAR($GbodyAddressChangeFileName);

print "** Opening: $GbodyDataLayoutFileName...\n";
        %LAYOUT                                 = &HASHloadFieldMap($GbodyDataLayoutFileName,$LGDEBUG_FTN);

unless($ADD_CHANGE_ON) {

        print "** Opening: $GmailingDBLayoutFileName...\n";
        %MDBLAYOUT = &HASHloadFieldMap($GmailingDBLayoutFileName,$LGDEBUG_FTN);

```

F:_Development\PRJ\PRJ_hbsp_extractOrder-WORKING\procOrder.pl
Printed at 19:48 on 06 Feb 1999

```
}

## open I/O
                                                                      {$eventType = 'ABORT'; $eventDesc = "MAIN::Cannot open GactivityLogFileName: \[$fileName\]"; &logE
if (!open(FH_log, ">>$GactivityLogFileName"))
vent(\*FH_log, $eventType, $eventDesc, $Guser, $LGDEBUG_FTN);}
                                                                      {$eventType = 'FAIL'; $eventDesc = "MAIN::Cannot open GoutFile: \[$GoutFile\]"; &logEvent(\*FH_log
if (!open(FH_out, ">$GoutFile"))
 $eventType, $eventDesc, $Guser, $LGDEBUG_FTN);}
                                                                      {$eventType = 'FAIL'; $eventDesc = "MAIN::Cannot open GreportFile: \[$GreportFile\]"; &logEvent(\*
if (!open(FH_report, ">$GreportFile"))
FH_log, $eventType, $eventDesc, $Guser, $LGDEBUG_FTN);}
                                                                      {$eventType = 'FAIL'; $eventDesc = "MAIN::Cannot open excReportFile: \[$GexcReportFile\]"; &logEve
if (!open(FH_excReport, ">$GexcReportFile"))
nt(\*FH_log, $eventType, $eventDesc, $Guser, $LGDEBUG_FTN);}
                                                                      {$eventType = 'FAIL'; $eventDesc = "MAIN::Cannot open excBODYReportFile: \[$GexcBODYReportFile\]";
if (!open(FH_excBODYReport, ">$GexcBODYReportFile"))
 &logEvent(\*FH_log, $eventType, $eventDesc, $Guser, $LGDEBUG_FTN);}
                                                                      {$eventType = 'FAIL'; $eventDesc = "MAIN::Cannot open GexceptionsFile: \[$GexceptionsFile\]"; &log
if (!open(FH_exc, ">$GexceptionsFile"))
Event(\*FH_log, $eventType, $eventDesc, $Guser, $LGDEBUG_FTN);}

## load %results fieldName keys from layout cf file (now in %LAYOUT)

@fieldKeys                  = undef;
@fieldKeys                  = &getFieldKeys(\%LAYOUT);

        ##print "--debug--\n"; foreach(@fieldKeys) { print "**".$_."| \n"; }

$numberRequiredFields   = scalar(@fieldKeys);

if(!defined($LAYOUT{'orderProcessingDSN'})) { $LAYOUT{'orderProcessingDSN'} = $GDSN; }

print "** Loading daily order data source: $LAYOUT{'orderProcessingDSN'}...\n";
@DB                         = &read_DSNtoAOH($LAYOUT{'orderProcessingDSN'},$LGDEBUG_FTN);

$totalRecords               = $#DB +1;
$GclientName                = $LAYOUT{'thisClientName'};

## startup OK event

$eventType = '_x_INFO'; $eventDesc = "procOrder R.$Grevision START OK - USER: $Guser/$GcfPath CLIENT: $GclientName  opDSN ($totalRecords): $LAYOUT{'orderP
rocessingDSN'} mdbDSN: $LAYOUT{'orderProcessingDSN'} -> $LAYOUT{'thisMailingTable'}  ($totalRecords recs) BODYdefn: $GbodyDataLayoutFileName"; &logEvent(\
*FH_log, $eventType, $eventDesc, $Guser, $LGDEBUG_FTN);

print "\n\n** Status...\n\n";
print "USER:\t\t$Guser \n";
print "CLIENT:\t\t$GclientName\n";
print "cfPATH:\t\t$GcfPath\n\n";
print "TOTAL RECORDS \t\t -> \"$LAYOUT{'orderProcessingDSN'}\":  $totalRecords\n";
unless($ADD_CHANGE_ON) { print "MAILING TABLE SOURCE \t -> \"$LAYOUT{'thisMailingDSN'} -> $LAYOUT{'thisMailingTable'}\"\n\n"; }
print "\n(Note: if this information is not correct, hit <CNTRL> - C to abort and correct <you have 5 seconds>)\n\n"; sleep 5;

print "** Begin processing...\n\n";

## flag duplicate e-mail addresses in order stream

%dupsMap = &checkDupsReturnMap(\@DB,$GDEBUG);

###################
#### main loop ####
###################

$excpCount                  = 0;
$warnCount                  = 0;
$outFileCount               = 0;
$exceptionsFileCount        = 0;
$excReportFileCount         = 0;
$reportFileCount            = 0;
$FUPcount                   = 0;
$reportLineCount            = 0;

for $mainCount (0 .. $#DB) {

```

```
$thisRecordID                   = $mainCount+1;
$eventRequiresFUP               = '';
$RECORD_VALID                   = 1;
$thisBuyStatus                  = '';
$thisShippingAddressBlock       = undef;
$thisBillingAddressBlock        = undef;
$thisMDBbillingAddressBlock     = undef;
$thisFullNumberedBody           = "\n";

## get relevant fields, normalized, this record

$thisFromAddress        = &normalizeSpacing(lc($DB[$mainCount]{'FromAddress'}));
$thisFromAddress        =~ s/[\t\r]/$Gspace/g;

$thisFromName           = &normalizeSpacing($DB[$mainCount]{'FromName'});
$thisFromName           =~ s/[\t\r]/$Gspace/g;

$thisSubject            = &normalizeSpacing(uc($DB[$mainCount]{'Subject'}));
$thisSubject            =~ s/[\t\r]/$Gspace/g;

$thisBody               = uc($DB[$mainCount]{'Body'});
$thisBody               =~ s/[\t\r]/$Gspace/g;

## split BODY into discrete lines

@BODYlines              = undef;
@topLines               = undef;
@bottomLines            = undef;

push(@BODYlines, $1) while ($thisBody =~ s/^([^\012\015]*)(\012\015?|\015\012?)//);

## skip lines that are the quoted BODY unless they're the order form -- example delimiters:
## $BODYstartToken = 'HARVARD BUSINESS SCHOOL PUBLISHING';
## $BODYendToken = 'FIRST NAME\:';

$BODYstartToken         = $LAYOUT{'startDelimiter'};
$BODYendToken           = $LAYOUT{'endDelimiter'};
$BODYstartToken         =~ s/[\t\n\r]//g;
$BODYendToken           =~ s/[\t\n\r]//g;
$startToken_ON          = 0;
$endToken_ON            = 0;
$thisBody               = undef;
$thisBodyTop            = undef;
$thisBodyBottom         = undef;
$lastLineExcluded       = undef;

for($count = 0; $count <= $#BODYlines; $count++) {

        $line                   = $BODYlines[$count];
        $thisFullNumberedBody   .= ('['.$count."] \t".$line."\n");

        unless($startToken_ON || $line =~ /^\>+/) {unless ($line !~ /\S+?/) {$thisBodyTop .= ($line.$Gpipe)}; push(@topLines,('['.$count.']  |'.$line.'|')
); }

                if(($line =~ /$BODYstartToken/) .. ($line =~ /$BODYendToken/)) {

                        if(!$startToken_ON && ($line =~ /$BODYstartToken/))     {$startToken_ON = 1;}
                        if(!$endToken_ON && ($line =~ /$BODYendToken/))         {$endToken_ON = 1; $thisBody .= ($line.$Gnewline); unless($line =~ /^\>+/
 || $line !~ /\S+?/ || $line =~ /[\[\]]+/) {$thisBodyBottom .= ($line.$Gpipe)} }

                next;
                }

        ## ignore bottom lines if part of the original -> indicators:  ">" || "[" || "]"

        if($startToken_ON && $endToken_ON) { unless($line =~ /^\>+/ || $line !~ /\S+?/ || $line =~ /[\[\]]+/) {$thisBodyBottom .= ($line.$Gpipe); push(@bo
ttomLines, ('['.$count.']  |'.$line.'|'));} }
        $thisBody .= ($line.$Gnewline);
}
```

F:_Development\PRJ\PRJ_hbsp_extractOrder-WORKING\procOrder.pl
Printed at 19:48 on 06 Feb 1999

Page 7

```

$thisBodyTop    =~ s/[\t\r]/__/g;
$thisBodyBottom =~ s/[\t\r]/__/g;

                    if($GDEBUG) { print "\n\n\n** DEBUG__THISBODY=".$thisBody."\n\n\n";}

## log/print warning if cannot find FOOTER ID BLOCK: CUSTNO|LISTID|MAILINGID|DIALOGID || if bad match yields the same for CUSTNO|LISTID

          $thisCustNO            =  -1;
          $thisListID            =  -1;
          $thisMailingID         =  -1;
          $thisDialogID          =  -1;
          $LISTCID_EXCEPT_ON     =   0;
          $MAILINGID_EXCEPT_ON   =   0;

          if($thisBody !~ /\[\[([^|]*?)\|([^|]*?)\|([^|]*?)\|?([^|]*?)\]\]/) {

                    $LISTCID_EXCEPT_ON = 1;

                    $eventType = 'EX_ERR'; $excpCount++; $eventDesc = " \<".$thisRecordID."\>   UNMATCHED FOOTER ID BLOCK: CUSTNO|LISTID|MAILINGID|DIALOGID";
&logEvent(\*FH_log, $eventType, $eventDesc, $Guser, $LGDEBUG_FTN);
                    $eventType = 'INFO'; $excpCount++; $eventDesc = " \<".$thisRecordID."\>   >>> ATTEMPTING SUBJECT LINE MAILING ID RECOVERY >>>"; &logEvent(
\*FH_log, $eventType, $eventDesc, $Guser, $LGDEBUG_FTN);

## test to see if there is a cf file synonymn (**,++, etc) for the MAILING ID which can be recovered in the absence of the footer token

                    if($thisMailingID == -1) { $thisMailingID = &subjectSynonymnLookup($thisSubject,\%LAYOUT); }
                    if($thisMailingID == -1) {

                              $RECORD_VALID=0;
                              $MAILINGID_EXCEPT_ON = 1;
                              $eventType = 'EX_ERR'; $excpCount++; $eventDesc = " \<".$thisRecordID."\>   MAILING ID RECOVERY WAS NOT SUCCESSFUL"; &logEvent(\*F
H_log, $eventType, $eventDesc, $Guser, $LGDEBUG_FTN);

                    }

                    if($thisMailingID != -1) {

                              $eventType = 'INFO'; $excpCount++; $eventDesc = " \<".$thisRecordID."\>   <<< MAILING ID RECOVERY SUCCESSFUL USING: $thisSubjectMa
ilingIDToken -> $LAYOUT{$thisSubjectMailingIDToken}"; &logEvent(\*FH_log, $eventType, $eventDesc, $Guser, $LGDEBUG_FTN);

                    }
          }

          else {

                    if($GDEBUG) { print "***DEBUG_  element1= $1 __ element2= $2 ___ element3= $3 ___ element4= $4\n"; sleep 1; }

                    $element1 = $1; $element2 = $2; $element3 = $3; $element4 = $4;

                    $thisCustNO            = ($element1 =~ /\S+?/) ? $element1 : -1;
                    $thisListID            = ($element2 =~ /\S+?/) ? $element2 : -1;
                    $thisMailingID         = ($element3 =~ /\S+?/) ? $element3 : -1;
                    $thisDialogID          = ($element4 =~ /\S+?/) ? $element4 : -1;

                    if ($thisCustNo == -1 && $thisListID == -1) {

                              $LISTCID_EXCEPT_ON = 1;
                              $MAILINGID_EXCEPT_ON = 1;
                              $eventType = 'EX_ERR'; $excpCount++; $eventDesc = " \<".$thisRecordID."\>   BAD MATCH:  CUSTNO, LISTID"; &logEvent(\*FH_log, $even
tType, $eventDesc, $Guser, $LGDEBUG_FTN);
                    }

                    ## log/print INFO if cannot find mailingID

                    if($thisMailingID == -1) { $eventType = 'INFO'; $eventDesc = " \<".$thisRecordID."\>   NO MAILING ID MATCH"; &logEvent(\*FH_log, $eventTyp
e, $eventDesc, $Guser, $LGDEBUG_FTN);}

                    ## log/print INFO if cannot find dialogID
```

```
                if($thisDialogID == -1) { $eventType = 'INFO'; $eventDesc = " \<".$thisRecordID."\>   NO DIALOG ID MATCH"; &logEvent(\*FH_log, $eventType,
 $eventDesc, $Guser, $LGDEBUG_FTN);}
        }

## check for /multi || quoted BUY TOKEN (eg: "YES")

        $buyTokenPosition       = 'none';
        $thisItemsSelectedStack = undef;
        @thisItemsSelected      = undef;

        unless($ADD_CHANGE_ON) {

                ## check for buy tokens, eg: YES in SUBJECT, TOP, BOTTOM || letter items: ABC || A,B,C || A-C

                        $buyTokenPosition = &detectBuyTokens($MULTI_BUY_ON,$Gsubj_buyToken,$thisSubject,$thisMailingID,$thisBodyTop,$thisBodyBottom,\%LAYO
UT,\@topLines,\@bottomLines,$GDEBUG);

                ## check for no-match case and except

                        if($buyTokenPosition eq 'none') { $eventRequiresFUP .= '.noTok' ; $FUPcount++; $eventType = 'ERR_FUP'; $warnCount++; $eventDesc =
" \<".$thisRecordID."\>   SUBJECT, TOP, BOTTOM -> NO BUY TOKEN MATCH ($Gsubj_buyToken)"; &logEvent(\*FH_log, $eventType, $eventDesc, $Guser, $LGDEBUG_FTN)
; }

                ## if match and MULTI_BUY_ON, prepare order items listing

                        if($buyTokenPosition ne 'none' && $MULTI_BUY_ON) {

                                $thisItemsSelectedStack = &extractDedupItemsSelectedStack(@thisItemsSelected);

                        }
        }

        ################
        ## INNER LOOP ##
        ################

        %results                = ();
        $countFields            = 0;
        $nullCriticalFields     = 0;
        $ADDRESS_ADEQUATE       = 1;

## process body fields by key; keys == standard field names; values == this BODY's field names mapped thereon

        foreach (@fieldKeys) {

        $key = $_;

        ## print "--DEBUG-- key = |$key| \n";

                $thisLayoutFieldName = uc($LAYOUT{$key});
                $thisLayoutFieldName =~ s/\s+$//;

                #####################################
                ###### MAIN MATCHING BLOCK ########
                #####################################

                $MATCH_DATA_ON = 0;
                $RULE = -1;

                $expectedMatch          = $thisLayoutFieldName.'\:.*?\[([^\[]+?)\]';
                $noBracketsMatch        = $thisLayoutFieldName.'\:.*?([^\012\015]+?)';

                CASE:   {

                        TRY_EXPECTED_MATCH:     if($thisBody =~ /$expectedMatch/)          { $MATCH_DATA_ON = 1; $matchElement = $1; $RULE = 1; last
 CASE; }
                        TRY_NO_BRAKETS_MATCH:   if($thisBody =~ /$noBracketsMatch/)        { $MATCH_DATA_ON = 1; $matchElement = $1; $RULE = 2; last
 CASE; }
```

```
                }
## A MATCH WAS FOUND, THIS KEY ... push into results set
                if($MATCH_DATA_ON) {

                        $countFields++;
                        $matchElement  =~ s/[^\w\s\-\.\/\#\!\@]/$Gspace/g;
                        $results{$key} =  &normalizeSpacing($matchElement);

## print "--DEBUG-- result = |$results{$key}| \n";

## build shipping and billing address blocks
                        CASE: {

                                if($key =~ /u[a-z]_/) {

                                        ## construct SHIPPING ADDRESSBLOCK for report

                                        if ($key =~ /[hijklmq]\_/) { $thisShippingAddressBlock .= (length($results{$key}) > 0)?($results{$key}.$Gpipe):'';
 }
                                        if ($key =~ /[efgnop]\_/) { $thisShippingAddressBlock .= (length($results{$key}) > 0)?($results{$key}.$Gspace):'';
 }

                                last CASE;
                                }

                                if($key =~ /[^u]_/) {

                                        ## construct BILLING ADDRESSBLOCK for report

                                        if ($key =~ /[hijklmq]\_/) { $thisBillingAddressBlock .= (length($results{$key}) > 0)?($results{$key}.$Gpipe):'';
}
                                        if ($key =~ /[efgnop]\_/) { $thisBillingAddressBlock .= (length($results{$key}) > 0)?($results{$key}.$Gspace):'';
}

                                last CASE;
                                }

                        }

                        ## check for null critical fields: any absence here will throw exception, unless CUSTNO

## print "--debug-- |$LAYOUT{'criticalFields'}| \n";sleep 1;

                        if ($key =~ /[$LAYOUT{'criticalFields'}]\_/ && $results{$key} !~ /\S+?/ && $thisCustNo < 1) {

                                $RECORD_VALID=0; $ADDRESS_ADEQUATE = 0;
                                $eventRequiresFUP .= (($eventRequiresFUP =~ /badAd/)?'':'.badAd'); $FUPcount++; $eventType = 'EX_FUP'; $excpCount++; $even
tDesc = " \<".$thisRecordID."\>   CRITICAL STANDARD FIELD_VALUE IS NULL: \[$thisLayoutFieldName\]"; &logEvent(\*FH_log, $eventType, $eventDesc, $Guser, $L
GDEBUG_FTN);

                        }

                }

## ALL MATCHES FAILED, THIS KEY

                if(!$MATCH_DATA_ON) {

                        $results{$key} = undef;
                        $eventType = 'EX_ERR'; $excpCount++; $eventDesc = " \<".$thisRecordID."\>   STANDARD FIELD_NAME=FIELD_VALUE PAIR UNPARSABLE: \[$th
isLayoutFieldName\]"; &logEvent(\*FH_log, $eventType, $eventDesc, $Guser, $LGDEBUG_FTN);

                # check to see is unparsable field is a critical field
                # critical fields -> from CF file: any absence here will throw exception

```

```
                        if ($key =~ /[$LAYOUT{'criticalFields'}]\_/) {

                                $RECORD_VALID=0; $ADDRESS_ADEQUATE = 0;
                                $eventRequiresFUP .= (($eventRequiresFUP =~ /badAd/)?'':'.badAd'); $FUPcount++; $eventType = 'EX_FUP'; $excpCount++; $even
tDesc = " \<".$thisRecordID."\>   UNPARSABLE STANDARD FIELD_NAME=FIELD_VALUE PAIR IS CRITICAL: \[$thisLayoutFieldName\]"; &logEvent(\*FH_log, $eventType,
$eventDesc, $Guser, $LGDEBUG_FTN);
                        }
                }

        }

        ########################
        #### END INNER LOOP ###
        ########################

######################
### SUMMARY RULES  ###
######################

## throw EX_ LISTNO/CUSTNO error exception if address is not OK && !phone, depending on CID_LID_REQUIRED from cf file

        if($LAYOUT{'CID_LID_REQUIRED'}) {

                if($LISTCID_EXCEPT_ON) {

                        $RECORD_VALID = 0;
                        $eventRequiresFUP .= (($eventRequiresFUP =~ /chkCLID/)?'':'.chkCLID'); $FUPcount++; $eventType = 'EX_FUP'; $excpCount++; $eventDes
c = " \<".$thisRecordID."\>   EITHER CID || LID ABSENT WHEN CF CRITICAL"; &logEvent(\*FH_log, $eventType, $eventDesc, $Guser, $LGDEBUG_FTN);

                }

        }

        if(!$LAYOUT{'CID_LID_REQUIRED'}) {

                if($LISTCID_EXCEPT_ON && !($ADDRESS_ADEQUATE && $results{'r\_phone'} !~ /\S+?/)) { $RECORD_VALID = 0; }

        }

## throw EX_ MAILINGID exception if cf file says it's required

        if($LAYOUT{'MAILINGID_REQUIRED'}) {

                if($MAILINGID_EXCEPT_ON) {

                        $RECORD_VALID = 0;
                        $eventRequiresFUP .= (($eventRequiresFUP =~ /chkMID/)?'':'.chkMID'); $FUPcount++; $eventType = 'EX_FUP'; $excpCount++; $eventDesc
= " \<".$thisRecordID."\>   MAILING ID ABSENT WHEN CF CRITICAL"; &logEvent(\*FH_log, $eventType, $eventDesc, $Guser, $LGDEBUG_FTN);

                }

        }

## throw INFO if less than number required

        if(($countFields < $numberRequiredFields)  && $thisCustNo < 1) {

                $eventType = 'INFO'; $eventDesc = " \<".$thisRecordID."\>   (OVERALL) NOT ALL STANDARD FIELDS FOUND: $countFields ($numberRequiredFields)"
; &logEvent(\*FH_log, $eventType, $eventDesc, $Guser, $LGDEBUG_FTN);
        }

## tag duplicates for FUP

if($dupsMap{$mainCount} =~ /\#\#\#.+\#\#\#/) { $eventRequiresFUP .= (($eventRequiresFUP =~ /dupAd/)?'':'.dupAd'); $FUPcount++; }

## catch thisBodyTop and thisBodyBottom blank yet address still extracted adequately (eg happens when cut & paste responses between the body delimiters)

if($RECORD_VALID && $thisBodyTop !~ /\S+?/ && $thisBodyBottom !~ /\S+?/) {

        $RECORD_VALID = 0;
        $eventType = 'EX_ERR'; $eventRequiresFUP .= '.chkBody' ; $FUPcount++; $excpCount++; $eventDesc = " \<".$thisRecordID."\>   CUSTOMER RESPONSE (Top,
 Bottom) NOT FOUND WHEN EXPECTED"; &logEvent(\*FH_log, $eventType, $eventDesc, $Guser, $LGDEBUG_FTN);

}

##################################################################
#### RECORD INVALID: Attempt recovery from mailing database ####
##################################################################

$RECORD_RECOVERED = 0;

if(!$RECORD_VALID && !$ADD_CHANGE_ON) {

        $eventType = 'INFO'; $eventDesc = " \<".$thisRecordID."\>   >>> BAD RECORD: ATTEMPTING RECOVERY-LOOKUP USING \"$thisFromAddress\", $LAYOUT{'thisMa
ilingDSN'} -> $LAYOUT{'thisMailingTable'}"; &logEvent(\*FH_log, $eventType, $eventDesc, $Guser, $LGDEBUG_FTN);

        %recoveredResults = undef;
        %recoveredResults = &querySELECT_DB(\%LAYOUT,'EMAIL',$thisFromAddress,$LGDEBUG_FTN);

## foreach(keys(%recoveredResults)) { print "=debug== key = $_ ____ value = $recoveredResults{$_} \n"; }

        CASE: {

                if($recoveredResults{'_exit_'} < 1) {

                        $RECORD_VALID = 0;
                        $eventType = 'EX_ERR'; $eventRequiresFUP .= (($eventRequiresFUP =~ /badMDB/)?'':'.badMDB') ; $FUPcount++; $excpCount++; $eventDesc
 = " \<".$thisRecordID."\>   MAILING TABLE RECOVERY-LOOKUP FAILED ON \"$thisFromAddress\""; &logEvent(\*FH_log, $eventType, $eventDesc, $Guser, $LGDEBUG_F
TN);

                last CASE;
                }
                if($recoveredResults{'_exit_'} == 1) {

                        $RECORD_RECOVERED = 1;
                        $RECORD_VALID = 1;
                        $eventType = 'WRN_FUP'; $eventRequiresFUP .= (($eventRequiresFUP =~ /goodMDB/)?'':'.goodMDB'); $FUPcount++; $excpCount++; $eventDe
sc = " \<".$thisRecordID."\>   <<< MAILING TABLE RECOVERY-LOOKUP SUCCESSFUL USING \"$thisFromAddress\""; &logEvent(\*FH_log, $eventType, $eventDesc, $Guse
r, $LGDEBUG_FTN);

                        ## construct $thisMDBbillingAddress

                        foreach $key(sort(keys(%MDBLAYOUT))) {

                                $value = $MDBLAYOUT{$key};

                        ## print "**DEBUG key = $key ___ value = $value \n";

                                if ($key =~ /[hijklmq]\_/) { $thisMDBbillingAddressBlock .= (length($recoveredResults{$value}) > 0)?($recoveredResults{$va
lue}.$Gpipe):''; }
                                if ($key =~ /[efgnop]\_/) { $thisMDBbillingAddressBlock .= (length($recoveredResults{$value}) > 0)?($recoveredResults{$val
ue}.$Gspace):''; }

                        }

                        ## restore mailingID and CID, if blank
## print "**DEBUG mailingID = $recoveredResults{'ED_MAILINGID'} __ custNo = $recoveredResults{'CID'} __ current custno = |$thisCustNo|\n";

                        if($thisMailingID !~ /\S+?/ || $thisMailingID == -1)    { $thisMailingID        = $recoveredResults{'ED_MAILINGID'}; }
                        if($thisCustNo !~ /\S+?/ || $thisCustNo == -1)          { $thisCustNo           = $recoveredResults{'CID'}; }

                last CASE;
                }

        }
```

```
}

#### add an empty flag where SHIPPING ADDRESS is returned blank

if($thisShippingAddressBlock !~ /\S+?/) { $thisShippingAddressBlock = '{ SAME }'; }
if($thisMDBbillingAddressBlock !~ /\S+?/) { $thisMDBbillingAddressBlock = '{ N/A }'; }

#### tag address block exceptions with diagnostic ques

if(!$RECORD_VALID) {

        CASE: {

                if($thisBillingAddressBlock !~ /\S+?/ && ($thisBodyTop =~ /\S+?/ || $thisBodyBottom =~ /\S+?/))        { $thisBillingAddressBlock = '=NO_
VALID_ADD/INSPECT='; last CASE;}
                if($thisBillingAddressBlock !~ /\S+?/ && $thisBodyTop !~ /\S+?/ && $thisBodyBottom !~ /\S+?/)        { $thisBillingAddressBlock = '{ BO
DY_EMPTY/INSPECT }'; last CASE; }
                if($thisBillingAddressBlock =~ /\S+?/ && !$ADDRESS_ADEQUATE)        { $thisBillingAddressBlock = 'CRIT
_FLD_NUL/INSPECT ->|'.$thisBillingAddressBlock.'|'; last CASE; }

                $thisBillingAddressBlock = '=UNKNOWN_ERR/INSPECT=|'.$thisBillingAddressBlock; last CASE;

        }

$eventRequiresFUP = ($eventRequiresFUP !~ /\S+?/) ? '.inspect' : $eventRequiresFUP;
}

############################################
## write valid record data to outfiles ##
############################################

$ED_ORDER_VALUE = 1;
$EXCEPTION_TYPE = 'OK';
$UPDATE_SUCCESSFUL = 0;
$THIS_KEY_FIELD = undef;
$THIS_KEY_FIELD_VALUE = undef;

        if($RECORD_VALID) {

                unless($ADD_CHANGE_ON) {

                        CASE: {

                                if(&queryUPDATE_DB(\%LAYOUT,$THIS_KEY_FIELD_VALUE = $thisFromAddress,$THIS_KEY_FIELD = "EMAIL","ED_ORDER",$ED_ORDER_VALUE,
"ED_MAILACTION",-1,$RECORD_RECOVERED,$LGDEBUG_FTN)) { $UPDATE_SUCCESSFUL=1; last CASE; };
                                if(&queryUPDATE_DB(\%LAYOUT,$THIS_KEY_FIELD_VALUE = $thisCustNo,$THIS_KEY_FIELD = "CID","ED_ORDER",$ED_ORDER_VALUE,"ED_MAI
LACTION",-1,$RECORD_RECOVERED,$LGDEBUG_FTN))        { $UPDATE_SUCCESSFUL=1; last CASE; };
                                if(&queryUPDATE_DB(\%LAYOUT,$THIS_KEY_FIELD_VALUE = $thisListID,$THIS_KEY_FIELD = "ED_LID","ED_ORDER",$ED_ORDER_VALUE,"ED_
MAILACTION",-1,$RECORD_RECOVERED,$LGDEBUG_FTN))      { $UPDATE_SUCCESSFUL=1; last CASE; };

                        }

                }

                if(!$UPDATE_SUCCESSFUL) {

                        $RECORD_VALID = 0;
                        $eventType = 'EX_ERR'; $eventRequiresFUP .= (($eventRequiresFUP =~ /badUpMDB/)?'':'.badUpMDB') ; $FUPcount++; $excpCount++; $event
Desc = " \<".$thisRecordID."\>   (MANUAL UPDATE REQUIRED) MAILING TABLE UPDATE FAILED ON ALL KEYS - LAST TRY: \"$THIS_KEY_FIELD -> $THIS_KEY_FIELD_VALUE\"
"; &logEvent(\*FH_log, $eventType, $eventDesc, $Guser, $LGDEBUG_FTN);

                }

        }

        if(!$RECORD_VALUE && !$UPDATE_SUCCESSFUL) {

                $eventType = 'EX_ERR'; $eventRequiresFUP .= (($eventRequiresFUP =~ /noUpMDB/)?'':'.noUpMDB') ; $FUPcount++; $excpCount++; $eventDesc = " \
<".$thisRecordID."\>   INVALID RECORD: MAILING TABLE NOT UPDATED"; &logEvent(\*FH_log, $eventType, $eventDesc, $Guser, $LGDEBUG_FTN);
```

```
        }

#### Buy tokens not found and not ADDRESS_CHANGE; move to EXC report

        unless($ADDRESS_CHANGE) { if($buyTokenPosition eq 'none') { $RECORD_VALID = 0; $ED_ORDER_VALUE = 2; } }

        if($RECORD_VALID) {

                $reportLineCount++;

                ## update MAILING TABLE


                ## write out files

                &writeRecord(\*FH_out,$GfieldDelimiter,($thisKey=&getNextDBkey_return($GoutFileDBkeysFileName,$LGDEBUG_FTN)),\%GoutFileRecord,'EXTENDED',
$LGDEBUG_FTN);
                &writeRecord(\*FH_report,$GfieldDelimiter,($thisKey=&getNextDBkey_return($GreportFileDBkeysFileName,$LGDEBUG_FTN)),\%GreportFileRecord,'NO
RMAL', $LGDEBUG_FTN);

                ## include ALL BODYs in the EXC_BODY file for potential review

                if($eventRequiresFUP =~ /\S+?/) { $EXCEPTION_TYPE = 'SOFT-EXC'; }

                        &writeRecord(\*FH_excBODYReport,$GfieldDelimiter,$thisBODYlogKey=&getNextDBkey_return($GexceptionsBODYDBkeysFileName,$LGDEBUG_FTN)
,\%GexceptionsBODYFileRecord,'BODY', $LGDEBUG_FTN);

        $outFileCount++;$reportFileCount++;
        }


################################################
## write exception record data to outfiles ##
################################################

        if(!$RECORD_VALID) {

                $EXCEPTION_TYPE = 'HARD-EXC';

                &writeRecord(\*FH_exc,$GfieldDelimiter,$thisKey=&getNextDBkey_return($GexceptionsDBkeysFileName,$LGDEBUG_FTN),\%GoutFileRecord,'EXTENDED',
$LGDEBUG_FTN);
                &writeRecord(\*FH_excReport,$GfieldDelimiter,$thisKey=&getNextDBkey_return($GexcReportDBkeysFileName,$LGDEBUG_FTN),\%GreportFileRecord,'NO
RMAL',$LGDEBUG_FTN);
                &writeRecord(\*FH_excBODYReport,$GfieldDelimiter,$thisBODYlogKey=&getNextDBkey_return($GexceptionsBODYDBkeysFileName,$LGDEBUG_FTN),\%Gexce
ptionsBODYFileRecord,'BODY', $LGDEBUG_FTN);

        $exceptionsFileCount++; $excReportFileCount++;
        }

print "\n";
}

########################
#### end main loop ####
########################

## print STDOUT exit state

$dateNow = &getDateStamp('WIN',$LGDEBUG_FTN);

print "\n** [procOrder version $Grevision] Done!  $totalRecords records processed by $Guser/$GcfPath for client: $GclientName **\n\n";
print "SUMMARY REPORT for: ".$dateNow."\n\n";
print "FILES WRITTEN \n";
printf  "\n%10d",$reportFileCount;
        print  " records written to REPORT FILE: $GreportFile";
printf  "\n%10d",$excReportFileCount;
        print  " records written to EXC_REPORT FILE: $GexcReportFile";
printf  "\n%10d",$outFileCount;
```

```
        print  " records written to OUTPUT FILE (all fields): $GoutFile";
printf  "\n%10d",$exceptionsFileCount;
        print  " records written to EXCEPTIONS FILE (all fields): $GexceptionsFile";
print "\n\nLOGGING\n\n";
print "All rule failure events for all time logged in detail to: $GactivityLogFileName\n";
printf  "\n%10d",$totalRecords;
print " BODYs for this session are keyed, categorized in: $GexcBODYReportFile, for review in an editor\n";

$eventType = '_X_INFO'; $eventDesc = "procOrder R.$Grevision EXIT OK - USER: $Guser/$GcfPath CLIENT: $GclientName  opDSN ($totalRecords): $LAYOUT{'orderPr
ocessingDSN'} mdbDSN: $LAYOUT{'orderProcessingDSN'} -> $LAYOUT{'thisMailingTable'}  ($totalRecords recs) BODYdefn: $GbodyDataLayoutFileName"; &logEvent(\*
FH_log, $eventType, $eventDesc, $Guser, $LGDEBUG_FTN);

close(FH_out); close(FH_exc); close(FH_log); close(FH_report); close(FH_excReport);close(FH_excBODYReport);

exit;

##################################################
#################### SUBS ######################
##################################################

#
# ----------------------------------------------------------------------
#

sub alphaOrdered {

my(@singleElementArray) = @_;

my $alpha       = 0;
my $i           = 0;

my $base = $singleElementArray[0];

for($i = 1; $i <= $#singleElementArray; $i++) {

        $ordBase = ord($base); $ordArr = ord($singleElementArray[$i]);

##      print "==debug=$i= base = |$base| ordBase= $ordBase ___ singleElementArray = $singleElementArray[$i] ordArr = $ordArr \n";

        if($ordBase > $ordArr) { return $alpha; }
        $base = $singleElementArray[$i];

}

$alpha = 1;

return $alpha;

}

#
# ----------------------------------------------------------------------
#
sub canonicalize {

my($inRecord) = @_;
my($elementsIndex, $fieldsIndex, $rec, $buf, $field) = undef;
my(@field) = undef;
my $space = ' ';
my $dashEscape = $space.'_d_esc_'.$space;

$inRecord = uc($inRecord);

my @allFields = undef;

@allFields = split(/$GfieldDelimiter/,$inRecord);
my $fieldsIndex = undef;

for($fieldsIndex=0; $fieldsIndex <= $#allFields; $fieldsIndex++) {
        if ($fieldsIndex != $emailFieldPos && $fieldsIndex != $fromAddressFieldPos) {
                if( $fieldsIndex ==  $companyFieldPos ||  $fieldsIndex == $lastNameFieldPos || $fieldsIndex == $firstNameFieldPos || $fieldsIndex == $titl
eFieldPos || $fieldsIndex == $countryFieldPos || $fieldsIndex == $departmentFieldPos  || $fieldsIndex == $cityFieldPos)  { $allFields[$fieldsIndex] =~ s/\
-/$dashEscape/g; }
                $allFields[$fieldsIndex] =~ s/\s+$//;                           # trailing space
                $allFields[$fieldsIndex] =~ s/^\s+//;                           # leading space
                $allFields[$fieldsIndex] =~ s/[\"\`\,\\\/\(\)]/$space/g;        # extraneous chars
                $allFields[$fieldsIndex] =~ s/\s{2,}/$space/g;                  # more than one separating space

        }
                elsif ($fieldsIndex == $emailFieldPos || $fieldsIndex != $fromAddressFieldPos) { $allFields[$fieldsIndex] =~ s/$space//g; }

        @field = split($space,$allFields[$fieldsIndex]);

        $field = undef;

        for($elementsIndex = 0; $elementsIndex <= $#field; $elementsIndex++) {

                CONVERT_TITLE_CASE: {

                $leaveUpper = 1;

                        if($fieldsIndex == $postalCodeFieldPos) { last CONVERT_TITLE_CASE; }
                        if($headerFields[$fieldsIndex] !~ /\w*IX\w*$/ && $field[$elementsIndex] =~ /^[^AEIOU\.][^AEIOU\.][^AEIOU\.]?$/) { last CONVERT_TIT
LE_CASE; }
                        if($field[$elementsIndex] =~ /^[A-Z]\.[A-Z]\.[A-Z]?\.?$/)   { last CONVERT_TITLE_CASE; }

                        unless($field[$elementsIndex] =~ /(AND)|(OR)|(NEW)|(UND)|(AT)|(CO)|(LTD)|(INC)|(OF)|(THE)|(RD)/) {

                                if($fieldsIndex == $firstNameFieldPos && ($field[$elementsIndex] =~ /^[A-Z][A-Z]$/ || $field[$elementsIndex] =~ /^[A-Z]\.?
[A-Z]\.[A-Z]?$/)) { last CONVERT_TITLE_CASE; }
                                if(($fieldsIndex == $IXFieldPos ||  $fieldsIndex == $companyFieldPos)  && ($field[$elementsIndex] =~ /^[A-Z]\.?[A-Z]\.?[A-
Z]$/ || $field[$elementsIndex] =~ /^[I][IV][I]*$/ || $field[$elementsIndex] =~ /^[^JS]\.?[A-Z]\.?$/ )) { last CONVERT_TITLE_CASE; }
                                if($fieldsIndex == $titleFieldPos && $field[$elementsIndex] =~ /^[ARHMGIVUCS][EMSQ\&TRAFIOPV][PADO\.]*$/ && $field[$elemen
tsIndex] !~ /[MD][RS]/)     { last CONVERT_TITLE_CASE; }
                                if($fieldsIndex == $countryFieldPos && ($field[$elementsIndex] =~ /^[A-Z]\.[A-Z]\.?[A-Z]?\.?$/ || $field[$elementsIndex] =
~ /^[A-Z][A-Z][A-Z]?$/) ) { last CONVERT_TITLE_CASE; }

                        }

                        unless($fieldsIndex == $stateFieldPos && length($field[$elementsIndex])>5) { if($fieldsIndex == $stateFieldPos || $fieldsIndex ==
$postalCodeFieldPos)    { last CONVERT_TITLE_CASE; } }

                $leaveUpper = 0;

                        $buf = lc($field[$elementsIndex]);

                        if($fieldsIndex != $emailFieldPos && $fieldsIndex != $fromAddressFieldPos)  { $buf = ucfirst($buf); last CONVERT_TITLE_CASE; }
                }

        if($leaveUpper) { $buf = $field[$elementsIndex]; }

        $field .= ($buf.$space);
        }

$rec .= ($field.$GfieldDelimiter);
}

chop($rec);

$rec =~ s/$dashEscape/\-/g;

return $rec;
}
```

```
#
# ----------------------------------------------------------------------------------------------------
#

sub checkDupsReturnMap {

my($ptrAOHDB,$GDEBUG)     = @_;
my(@DB)                   = @$ptrAOHDB;

## prepare map of duplicates on FromAddress

my %dupsMap               = undef;
my $j                     = undef;

for $j (0 .. $#DB) {

        $dupsMap{$j} = $DB[$j]{'FromAddress'}

}

my @mirrorDupsMap         = undef;
@mirrorDupsMap            = values(%dupsMap);
my $key                   = undef;
my $value                 = undef;
my @arr                   = undef;
my $number                = undef;

foreach $key(keys %dupsMap) {

        $value          = $dupsMap{$key};
        next if($value !~ /\S+?/);

        @arr            = undef;
        @arr            = grep /$value/, @mirrorDupsMap; .
        $number         = scalar(@arr);

        if($number > 1) {

                $eventType      = 'INFO'; $eventDesc = "DUPLICATE FromAddress ITEM DETECTED ON IMPORT & FLAGGED: ($number|$key|$value)"; &logEvent(\*FH_lo
g, $eventType, $eventDesc, $Guser, $LGDEBUG_FTN);
                $dupsMap{$key}  = ('###'.$number.'### '.$dupsMap{$key});

        ## print $dupsMap{$key}."\n";

        }
        ## print "** key=".$key." *** value=".$dupsMap{$key}."___occurrences=".$number."\n";
}

return %dupsMap
}

#
# ---------------------------------------------------------------------
#

sub checkMultiBuy_Extract {

my($thisLine,$thisMailingID,$pos,$ptrLAYOUT,$GDEBUG) = @_;
my($exit) = 0;

my(%LAYOUT)                       = %$ptrLAYOUT;
my(@rangeStack)                   = undef;
my(@singleStack)                  = undef;
my($countRangeMatches)            = 0;
my($repeat)                       = 1;
my($match)                        = undef;
my($replacementEscChar)           = '%';
my($preProcRemoveChar)            = '^';
my($elementSeparator)             = ' ';
```

```

## PREPARE THE LINE FOR MATCHING

$thisLine                        = uc($thisLine);

## RUN RULESET FROM RULES CF FILE

foreach(@GPreProcRules) { eval; }

my $thisPreExtractionLine        = $thisLine;
my @testSingleStack              = undef;
my $maxASCII                     = 0;

## EXTRACTION LOOP

while($repeat) {

my $RULE = 0;
my $match = undef;

##print "\n\n--debug-- maxASCII = $maxASCII -- thisLine IS CURRENTLY |$thisLine| \n\n";

        MATCH_CASE: {

                SINGLE_CASE: {

##                      if($thisLine =~ /^([A-Z])$/)                                                      { $match = $1; $RULE = -1;  last MATCH_CASE; }
                        if($thisLine =~ /^([A-Z])[^A-Z\^\%\@\-]{1}/)                                      { $match = $1; $RULE = -2;  last MATCH_CASE; }
                        if($thisLine =~ /[^A-Z\^\%\@\-]([A-Z])[^A-Z\%\^\@\-]{1}[^A-Z\^\%\@\-]*/)          { $match = $1; $RULE = -3;  last MATCH_CASE; }

                }

                RANGE_CASE: {

                        if($thisLine =~ /^([A-Z]\-[A-Z])[^A-Z\^\%\@]{1}/)                                 { $match = $1; $RULE = 1; last MATCH_CASE; }
                        if($thisLine =~ /[^A-Z\%\^\@]{1}([A-Z]\-[A-Z])[^A-Z\^\%\@]{1}/)                   { $match = $1; $RULE = 2; last MATCH_CASE; }
                        ## if($thisLine =~ /([A-Z][A-Z][A-Z]*)[^A-Z\^\%]{1}/)                             { $match = $1; $RULE = 3; last MATCH_CASE; }
                        ## if($thisLine =~ /[^A-Z]{1}([A-Z][A-Z][A-Z]*)/)                                 { $match = $1; $RULE = 4; last MATCH_CASE; }
                }



        $repeat = 0;
        }

        $thisLine =~ s/$match/$replacementEscChar/g;

        $thisCheckedMailingID = ($thisMailingID!=0) ? $thisMailingID : -1;

                ##print "\n- debug - ELEMENT CONSIDERED with break = $RULE |$match| \n";
                if($GDEBUG) { print "--debug-- BEGIN RULES -- RANGE = |$LAYOUT{$thisCheckedMailingID}| from $thisCheckedMailingID \n"; }

## BEGIN RULES TO SEE WHAT WE GOT ##

        $PUSH = 1;

        if($RULE > 0) {

                my(@arr) = undef;
                my $tempMatch = $match;
                $tempMatch =~ s/\-//g;
                @arr = split(//,$tempMatch);

                if(!&alphaOrdered(@arr)) {

                        ## print "-- debug -- REJECT - not alpha match |$match| in line |$thisPreExtractionLine| \n";
                        $PUSH = 0;

                }

        }

        #if($RULE > 0 && ord($arr[0]) <= $maxASCII) {

#               print "-- debug -- REJECT - range case |$match| out of alpha order with current stack!\n";
#               if(ord($match)>$maxASCII) {$maxASCII = ord($match); }
#               $PUSH = 0;

        #}
        if(defined($match) && $match !~ /[$LAYOUT{$thisCheckedMailingID}]/) {

                ## print "--debug-- REJECT - not in RANGE = |$LAYOUT{$thisCheckedMailingID}| match |$match| in line |$thisPreExtractionLine|  \n" ;
                if(ord($match)>$maxASCII) {$maxASCII = ord($match); }
                $PUSH = 0;

        }

        if($PUSH && $RULE > 0 && $match =~ /\S+?/) {

##              push(@rangeStack,  ('<'.$RULE.'> |'.$match)); $exit = 1;
                $match =~ s/\-/ thru /g;
                push(@rangeStack, $match);
                $exit=1;
        }

        if($PUSH && $RULE < 0 && $match =~ /\S+?/) {

                push(@testSingleStack,$match);

                unless(ord($match) <= $maxASCII) {

##                      push(@singleStack, ('<'.$RULE.'> |'.$match));
                        push(@singleStack,$match);
                        $exit=1;

                }
                        if(ord($match)>$maxASCII) {$maxASCII = ord($match); }
        }

                if($GDEBUG) { if(!defined($match)) {print "--debug-- match not defined! in line= |$thisPreExtractionLine|\n"; }}
}

## consolidate extractions, pre report

foreach(@rangeStack)    { unless($_ !~ /\S+?/) { push(@thisItemsSelected,$_); }}
foreach(@singleStack)   { unless($_ !~ /\S+?/) { push(@thisItemsSelected,$_); }}

## SPECIAL POST EXTRACTION CASE RULES

        my $FIND_BUY_ALL = 0;

        foreach(@GBuyAllIndicators) { s/[\n\r]//g; next if($_ !~ /\S+?/); if($thisPreExtractionLine =~ /$_/) { $FIND_BUY_ALL = 1; } }

        if ($FIND_BUY_ALL) {

                if($exit) {

                        my $t = '.vfyTok';
                        $eventType = 'WRN_FUP'; ($eventRequiresFUP !~ /$t/) ? $eventRequiresFUP .= $t :  $eventRequiresFUP ; $FUPcount++; $excpCount++; $e
ventDesc = " \<".$thisRecordID."\>   checkMultiBuy_Extract :: AMBIGUOUS ALL-ITEM SELECTION DETECTED IN EXTRACTION LOCATION <$pos>"; &logEvent(\*FH_log, $e
ventType, $eventDesc, $Guser, $LGDEBUG_FTN);

                }

                push(@thisItemsSelected," ALL ($thisMailingID) "); $exit=1;
        }

        ## DETECT DIFFERENCE IN EXTRACTION STACK AND MAX EXTRACTION STACK

        if ($exit && (scalar(@singleStack) != scalar(@testSingleStack)) ) {

                my $t = '.vfyTok';
                $eventType = 'WRN_FUP'; ($eventRequiresFUP !~ /$t/) ? $eventRequiresFUP .= $t:  $eventRequiresFUP ; $FUPcount++; $excpCount++; $eventDesc
 = " \<".$thisRecordID."\>   checkMultiBuy_Extract :: AMBIGUOUS EXTRACTION WARNING / INSPECT IN EXTRACTION LOCATION <$pos>"; &logEvent(\*FH_log, $eventType
 , $eventDesc, $Guser, $LGDEBUG_FTN);

        }

        my $FIND_ADD_CHANGE = 0;

        foreach(@GAddressChangeIndicators) { s/[\n\r]//g; next if($_ !~ /\S+?/); if($thisPreExtractionLine =~ /$_/) { $FIND_ADD_CHANGE = 1; } }

        if ($FIND_ADD_CHANGE) {

                my $t = '.chkAC';
                $eventType = 'WRN_FUP'; ($eventRequiresFUP !~ /$t/) ? $eventRequiresFUP .= $t :  $eventRequiresFUP ; $FUPcount++; $excpCount++; $eventDesc
 = " \<".$thisRecordID."\>   checkMultiBuy_Extract :: POSSIBLE ADD/CHANGE REQUEST DETECTED IN EXTRACTION LOCATION <$pos>"; &logEvent(\*FH_log, $eventType,
$eventDesc, $Guser, $LGDEBUG_FTN);

        }

        my $FIND_QUESTION = 0;

        if($thisPreExtractionLine =~ /\?\s/) { $FIND_QUESTION = 1; }

        if ($FIND_QUESTION) {

                my $t = '.chkQst';
                $eventType = 'WRN_FUP'; ($eventRequiresFUP !~ /$t/) ? $eventRequiresFUP .= $t :  $eventRequiresFUP ; $FUPcount++; $excpCount++; $eventDesc
 = " \<".$thisRecordID."\>   checkMultiBuy_Extract :: POSSIBLE ADD/CHANGE OR CARE REQUEST DETECTED IN EXTRACTION LOCATION <$pos>"; &logEvent(\*FH_log, $ev
entType, $eventDesc, $Guser, $LGDEBUG_FTN);

        }

        my $FIND_AI_ONLY = 0;

        if ($exit && scalar(@singleStack) == 2 && scalar(@testSingleStack) == 2) {

        ## $x = scalar(@singleStack);
        ## $y = scalar(@testSingleStack);

        ##print "-- debug -- AI TEST ___ singleStackLen = $x __ testSingleStackLen = $y \n";

                if(grep /(A)|(I)/, @singleStack) { $FIND_AI_ONLY = 1;}

        }

        if ($FIND_AI_ONLY) {

                my $t = '.vfyTok';
                $eventType = 'WRN_FUP'; ($eventRequiresFUP !~ /$t/) ? $eventRequiresFUP .= $t :  $eventRequiresFUP ; $FUPcount++; $excpCount++; $eventDesc
 = " \<".$thisRecordID."\>   checkMultiBuy_Extract :: AMBIGUOUS EXTRACTION WARNING: \"A\" or \"I\" ONLY DETECTED"; &logEvent(\*FH_log, $eventType, $eventD
esc, $Guser, $LGDEBUG_FTN);

        }

        my $LIKELY_PARSE_ERRORS = 0;

        if ($exit && $thisPreExtractionLine =~ /[\[\]]+/) { $LIKELY_PARSE_ERRORS = 1; }

        if ($LIKELY_PARSE_ERRORS) {

                my $t = '.vfyTok';
                $eventType = 'WRN_FUP'; ($eventRequiresFUP !~ /$t/) ? $eventRequiresFUP .= $t :  $eventRequiresFUP ; $FUPcount++; $excpCount++; $eventDesc
 = " \<".$thisRecordID."\>   checkMultiBuy_Extract :: AMBIGUOUS EXTRACTION WARNING: BRACKETS FOUND IN EXTRACTION LOCATION <$pos>"; &logEvent(\*FH_log, $ev
entType, $eventDesc, $Guser, $LGDEBUG_FTN);
```

```
        }


        ##print "\n-$thisRecordID- debug singleStack -- \n";
        ##print @singleStack;
        ##print "\n-- debug testSingleStack -- \n";
        ##print @testSingleStack;
        ##print "\n-$thisRecordID---------------------------\n";


return $exit;
}

#
# -------------------------------------------------------------------
#

sub detectBuyTokens {

my($MULTI_BUY_ON,$Gsubj_buyToken,$thisSubject,$thisMailingID,$thisBodyTop,$thisBodyBottom,$ptrHASHLAYOUT,$ptrSCALARtopLines,$ptrSCALARbottomLines,$GDEBUG)
  = @_;

my %LAYOUT              = %$ptrHASHLAYOUT;
my @topLines            = @$ptrSCALARtopLines;
my @bottomLines         = @$ptrSCALARbottomLines;
my $buyTokenPosition    = 'none';
$thisBuyStatus          = undef;
@thisItemsSelected      = undef;
$thisItemsSelectedStack = undef;

my $exit = 0;

## print "--debug-- buyToken= |$Gsubj_buyToken|\n";

FIND_BUY_TOKENS: {

##
## &checkMultiBuy_Extract returns success on a good match/extraction;
## @thisItemsSelected is created/populated at this time as well
##

## check $thisSubject

## print "\n\n==========debug-- SUBJECT = |$thisSubject| \n";

        if(!$MULTI_BUY_ON && $thisSubject =~ /$Gsubj_buyToken/) {

                $buyTokenPosition = ' S '; $thisBuyStatus = ('<'.$buyTokenPosition.'> |'.$thisSubject.'|');

## print "--debug-- thisBuyStatus= |$thisBuyStatus| \n";

        last FIND_BUY_TOKENS;
        }

        if($MULTI_BUY_ON) {

                if(&checkMultiBuy_Extract($thisSubject,$thisMailingID,' S ',\%LAYOUT,$GDEBUG)) {

                        $buyTokenPosition = ' S '; $thisBuyStatus = '<'.$buyTokenPosition.'>';

                last FIND_BUY_TOKENS;
                }

        }

## check @topLines
## print "\n\n==========debug-- thisBodyTop = |$thisBodyTop| \n";

```

```
        if(!$MULTI_BUY_ON) {

                foreach(@topLines) {

                        $thisTopLine = $_;

                        if($thisTopLine =~ /$Gsubj_buyToken/) {

                                $buyTokenPosition = ' T ';
                                $thisBuyStatus = '<'.$buyTokenPosition.'> '.$thisTopLine;

## print "--debug-- thisBuyStatus= |$thisBuyStatus| \n";

                        }

                }

        last FIND_BUY_TOKENS;
        }


        if($MULTI_BUY_ON) {

                if(&checkMultiBuy_Extract($thisBodyTop,$thisMailingID,' T ',\%LAYOUT,$GDEBUG)) {

                        $exit =1;

                        $buyTokenPosition = ' T ';
                        $thisBuyStatus = '<'.$buyTokenPosition.'> ';

                last FIND_BUY_TOKENS;
                }

        }


## check @bottomLines

## print "\n\n==========debug-- thisBodyBottom = |$thisBodyBottom| \n";

        if(!$MULTI_BUY_ON) {

                foreach(@bottomLines) {

                        $thisBottomLine = $_;

                        if($thisBottomLine =~ /$Gsubj_buyToken/) {

                                $buyTokenPosition = ' B ';
                                $thisBuyStatus = '<'.$buyTokenPosition.'> '.$thisBottomLine;

## print "--debug-- thisBuyStatus= |$thisBuyStatus| \n";

                        }

                }

        last FIND_BUY_TOKENS;
        }

        if($MULTI_BUY_ON) {

                if(&checkMultiBuy_Extract($thisBodyBottom,$thisMailingID,' B ',\%LAYOUT,$GDEBUG)) {

                        $exit =1;

                        $buyTokenPosition = ' B ';
                        $thisBuyStatus = '<'.$buyTokenPosition.'> ';

```

```
                    last FIND_BUY_TOKENS;
                    }

          }

}

## print "--debug-- ==== $i ====== buyTokenPosition= |$buyTokenPosition|\n";

return $buyTokenPosition;
}

#
# ----------------------------------------------------------------------
#

sub extractDedupItemsSelectedStack {

my(@thisItemsSelected) = @_;

my $thisItemsSelectedStack = undef;
my $thisItemsSelectedStackSeparator = ',';
my $i = undef;

for($i = 0; $i <= $#thisItemsSelected; $i++) {

          my $item = $thisItemsSelected[$i]; # all items must be normalized!

          next if($item !~ /\S+?/);

          $item=&normalizeSpacing($item);

          $thisItemsSelectedStack .= ($item.$thisItemsSelectedStackSeparator);

}

$thisItemsSelectedStack =~ s/\,\s*$//;

return  $thisItemsSelectedStack;
}

#
# ----------------------------------------------------------------------
#

sub getDateStamp {

# get standardized date stamp (UNIX, WINNT)
# returns a date stamp

my($Gplatform,$LGDEBUG_FTN) = @_;

if(!defined($Gplatform)) { $Gplatform = 'WIN'; }
my $GTIME           = '';
my $day             =  undef;
my $val             =  undef;

if ($Gplatform ne 'WIN') { $val = `date '+%y%m%d%H%M%S-%a'`; }
          else {
                    $val = `date /t`;
                    chop($val);
                    $val =~ /(\w+)\s+(\d+)\/(\d+)\/(\d{2})(\d{2})/;
                    $val = $5.$2.$3; $day = $1;
          }

if ($Gplatform eq 'WIN') {

                    $GTIME = `echo '' | time`;
                    $GTIME =~ /\:\s*(\d{1,2})\:(\d{2})\:(\d{2})\.(\d{2})/;
```

```
                my $leadingDigit = $1;
                if (length($leadingDigit) == 1) { $leadingDigit = ('0'.$leadingDigit); }
                $val .= $leadingDigit.$2.$3.$4.'-'.$day;
        }

if($LGDEBUG_FTN) { print "DEBUG__gdate=".$val."\n"; }

return $val;

}

#
# ----------------------------------------------------------------------
#

sub getFieldKeys {

my($ptrLayout) = @_;
my(%LAYOUT) = %$ptrLayout;

## prepare and count required fields from body data layout for check against body

my(@fieldKeys)  = undef;
my($k)          = undef;

##$f = scalar(@fieldKeys); print "--debug-- scalarStartFtn=$f \n";

foreach $k(sort keys %LAYOUT) {

        next if($k !~ /^[a-z]{1,2}_/);  ## these elements are not fieldnames but are other CF parameters (name \t value)

        ## print "--debug-- k= |$k|\n";

        if($k =~ /\S+?/) { push(@fieldKeys,$k); }
}

shift @fieldKeys;
## $f = scalar(@fieldKeys); print "--debug-- scalarEndFtn=$f \n";

return @fieldKeys;
}

# -----------------------------------------------------------------------------------------------

sub getKeyPos_Outfile {

my($ptrHASHrecordTemplate) = @_;

my(%HASHrecordTemplate) = %$ptrHASHrecordTemplate;

my @record                      = undef;
$salutationFieldPos     = -1;
$IDFieldPos             = -1;
$LIDFieldPos            = -1;
$MailingIDFieldPos      = -1;
$firstNameFieldPos      = -1;
$lastNameFieldPos       = -1;
$IXFieldPos             = -1;
$titleFieldPos          = -1;
$stateFieldPos          = -1;
$countryFieldPos        = -1;
$postalCodeFieldPos     = -1;
$emailFieldPos          = -1;
$companyFieldPos        = -1;
$addressFieldPos        = -1;
$departmentFieldPos     = -1;
$cityFieldPos           = -1;
$fromAddressFieldPos    = -1;

```

```
my $i                    = 0;

@record = sort(keys(%HASHrecordTemplate));

##print "---DEBUG---"; print @record; sleep 4;

for($i=0; $i <= $#record; $i++) {

        $record[$i] = uc($record[$i]);

        CASE: {

                if ($record[$i] =~ /FROM\_ADDRESS/)                                               { $fromAddressFieldPos    = $i; last CASE; }
                if ($record[$i] =~ /(EMAIL)|(E-MAIL)|(INTERNET_ADDR)|(INTERNET_AD)|(INTERNETAD)/) { $emailFieldPos          = $i; last CASE; }
                if ($record[$i] =~ /(COUNTRY)|(CTRY)/)                                            { $countryFieldPos        = $i; last CASE; }
                if ($record[$i] =~ /(ZIP)|(POSTAL)/)                                              { $postalCodeFieldPos     = $i; last CASE; }
                if ($record[$i] =~ /(INST)|(COMPANY)|(ORGANIZATION)/)                             { $companyFieldPos        = $i; last CASE; }
                if ($record[$i] =~ /DEPARTMENT/)                                                  { $departmentFieldPos     = $i; last CASE; }
                if ($record[$i] =~ /(ADDR)|(ADDRESS)/)                                            { $addressFieldPos        = $i; last CASE; }
                if ($record[$i] =~ /CITY/)                                                        { $cityFieldPos           = $i; last CASE; }
                if ($record[$i] =~ /(CUSTNO)/)                                                    { $IDFieldPos             = $i; last CASE; }
                if ($record[$i] =~ /(LISTID)/)                                                    { $LIDFieldPos            = $i; last CASE; }
                if ($record[$i] =~ /(PRIORITY)|(MAILING)/)                                        { $MailingIDFieldPos      = $i; last CASE; }
                if ($record[$i] =~ /(.*F\w+NAME.*)|(FIRST)/)                                      { $firstNameFieldPos      = $i; last CASE; }
                if ($record[$i] =~ /(.*L\w+NAME.*)|(LAST)/)                                       { $lastNameFieldPos       = $i; last CASE; }
                if ($record[$i] =~ /^.*IX.*$/)                                                    { $IXFieldPos             = $i; last CASE; }
                if ($record[$i] =~ /TITLE/)                                                       { $titleFieldPos          = $i; last CASE; }
                if ($record[$i] =~ /STATE/ || $record[$i] =~ /^\_C$/)                             { $stateFieldPos          = $i; last CASE; }
        }
}

if ($firstNameFieldPos  == -1)          { print STDOUT "[HEADER] WARNING: \/\\w\+F\\w\+NAME\/ was not found!\n"; }
if ($lastNameFieldPos   == -1)          { print STDOUT "[HEADER] WARNING: \/\\w\+L\\w\+NAME\/ was not found!\n"; }
if ($firstNameFieldPos  == -1)          { print STDOUT "[HEADER] WARNING: \/\\w\*IX\\w\*NAME\/ was not found!\n"; }
if ($titleFieldPos      == -1)          { print STDOUT "[HEADER] WARNING: \/TITLE\/ was not found!\n"; }
if ($stateFieldPos      == -1)          { print STDOUT "[HEADER] WARNING: \/STATE\/ was not found!\n"; }
if ($postalCodeFieldPos == -1)          { print STDOUT "[HEADER] WARNING: \/ZIP|POSTAL\/ was not found!\n"; }
if ($countryFieldPos    == -1)          { print STDOUT "[HEADER] WARNING: \/(COUNTRY)|(CTRY)\/ was not found!\n"; }
if ($emailFieldPos      == -1)          { print STDOUT "[HEADER] WARNING: \/(EMAIL)|(E-MAIL)|(INTERNET_ADDR)|(INTERNET_AD)|(INTERNETAD)\/ was not found!\n
"; }

}


#
# ----------------------------------------------------------------------
#

sub getNextDBkey_return {

my($keyFileName,$LGDEBUG_FTN) = @_;

$LOCK_SH = 1;
$LOCK_EX = 2;
$LOCK_NB = 4;
$LOCK_UN = 8;

$POS_BOF = 0;
$POS_CUR = 1;
$POS_EOF = 2;

$/ = undef;
$startKeys = 0;

if(! -s $keyFileName) { `echo $startKeys >> $keyFileName`; }

OPEN_DB: if (!open(KEYS, "+< $keyFileName")) {$exit = 0; $eventType = 'FAIL'; $eventDesc = "getNextDBkey_return::Cannot open: \[$keyFileName \]"; &logEven
t(\*FH_log, $eventType, $eventDesc, $Guser, $LGDEBUG_FTN);}

```

```
flock(KEYS, $LOCK_EX);

        my($count)      = <KEYS>;
        $nextKey        = $count+1;
        seek(KEYS,0,$POS_BOF);
        print KEYS $nextKey;

flock(KEYS, $LOCK_UN);
close(KEYS);

return $nextKey;
}

#
# ----------------------------------------------------------------------
#

sub HASHloadFieldMap {

my($fileName,$LGDEBUG_FTN) = @_;

if (!open(FH, "$fileName")) { $eventType = 'FAIL'; $eventDesc = "HASHloadFieldMap::Cannot open: \[$keyFileName \]"; &logEvent(\*FH_log, $eventType, $event
Desc, $Guser, $LGDEBUG_FTN);}

$/ = "\n";

my(@fieldMapTemp) = <FH>;
my($key) = undef;
my($value) = undef;
my(%fieldMap) = undef;

close(FH);

        foreach (@fieldMapTemp) {

                my $item = $_;

                next if(($item =~ /\/\//) || ($item !~ /\S+?/)); # comments

                ($key, $value) = split(/[\t\.]/,$item);

                $value =~ s/[\n\r]//g;

                $fieldMap{$key} = $value;

        }

return %fieldMap;
}

#
# ----------------------------------------------------------------------
#

sub loadFile_SCALAR {

my($fileName) = @_;

if (!open(FILE, "$fileName")) {$eventType = 'FAIL'; $eventDesc = "loadArray::Cannot open fileName: \[$fileName\]"; &logEvent(\*FH_log, $eventType, $eventD
esc, $Guser, $LGDEBUG_FTN);}

$/ = "\n";

my(@all) = undef;

while(<FILE>) {

        my $buf = $_;
        next if($buf !~ /\S+?/ || $buf =~ /^\/\//);
        push(@all, $buf);

}

return @all;
}

#
# -------------------------------------------------------------------------
#

sub logEvent {

my($FH,$eventType,$eventDesc,$Guser,$LGDEBUG_FTN) = @_;

my $dateNow = &getDateStamp('WIN',$LGDEBUG_FTN);
# print event to STDOUT

if($eventType eq 'ABORT') {

        $foo = `mapisend -u administrator -p password -r aestes\@e-dialog.com -s "** $dateNow procOrder - Critical Error"  -m "$Guser $GclientName $eventT
ype $eventDesc"`;
        die "Error! Cannot log! ".$eventType."\t".$eventDesc."\n" ;
}

if($eventType !~ /_x_/) { print $eventType."\t".$eventDesc."\n"; }

        else { $eventType =~ s/_x_//; }

# write event to activity log

        &writeRecord(\*FH_log,$GfieldDelimiter,$thisLogKey=&getNextDBkey_return($GactivityLogKeysFileName,$LGDEBUG_FTN),\%GactivityLogRecord,'',$LGDEBUG_F
TN);
        if($eventType eq 'FAIL') {

                $foo = `mapisend -u administrator -p password -r aestes\@e-dialog.com -s "** $dateNow procOrder - Critical Error"  -m "$Guser $GclientName
 $eventType $eventDesc"`;
                exit;

        }
}

#
# -------------------------------------------------------------------------
#

sub normalizeSpacing {

my($dirty) = @_;

$dirty =~ s/\s{2,}/$space/g; $dirty =~ s/\s+$//; $dirty =~ s/^\s+//;
$clean = $dirty;

return $clean
}

#
# -------------------------------------------------------------------------
#

sub queryUPDATE_DB {

my($ptrLAYOUT,$thisQueryKey, $thisQueryFieldName, $fieldName, $fieldValue, $fieldName1, $fieldValue1, $RECORD_RECOVERED, $LGDEBUG_FTN) = @_;

##print "** DSN = $LAYOUT{'thisMailingDSN'} \n";

my(%LAYOUT) = %$ptrLAYOUT;
my $RS = undef;
my $Conn = undef;
my $Errors = undef;
my @fieldNames = undef;
my $PID = -1;
my $recCount = 0;
my $EXIT = 1;

# construct/open DSN

my $Conn = undef;

$Conn = new Win32::OLE("ADODB.Connection");
$Conn->Open($LAYOUT{'thisMailingDSN'},"","");

my(%DB) = undef;

# Load data into recordset

        my $mailingTable = $LAYOUT{'thisMailingTable'};
        my %foo = undef;

# look to see if update record exists, $thisQueryKey

        unless($RECORD_RECOVERED) {

                my $RECORD_EXISTS = 1;
                %foo = &querySELECT_DB(\%LAYOUT,$thisQueryFieldName,$thisQueryKey,$LGDEBUG_FTN);
                if($foo{'_exit_'} < 1) { $RECORD_EXISTS = 0; $EXIT = 0; return $EXIT; }
        }

        $thisQuery = "UPDATE $mailingTable SET $fieldName = \'$fieldvalue\', $fieldName1 = \'$fieldvalue1\'  WHERE $thisQueryFieldName = \'$thisQueryKey\'
";

        $RS = $Conn->Execute($thisQuery);

##%foo = %$RS;
##foreach(keys(%foo)) { print "---key = $_ __ value = $foo{$_} \n"; }

        if(!$RS) {

                $Errors = $Conn->Errors();
                print "Errors:\n";
                foreach $error (keys %$Errors) {
                        print $error->{Description}, "\n";
                }
                $eventType = 'WRN_FUP'; $eventRequiresFUP .= (($eventRequiresFUP =~ /badMDB/)?'':'.badMDB'); $eventDesc = " \<".$thisRecordID."\>   ($LAYO
UT{'thisMailingDSN'}) queryUPDATE_DB :: Cannot create RS: \[$thisQuery\]"; &logEvent(\*FH_log, $eventType, $eventDesc, $Guser, $LGDEBUG_FTN);

        $EXIT = 0;
        return $EXIT;
        }

        $eventType = 'INFO'; $eventDesc = " \<".$thisRecordID."\>   ($LAYOUT{'thisMailingDSN'}) queryUPDATE_DB :: UPDATE SUCCESSFUL: \[$thisQuery\]"; &log
Event(\*FH_log, $eventType, $eventDesc, $Guser, $LGDEBUG_FTN);

return $EXIT;
}

#
# ----------------------------------------------------------------------
#

sub querySELECT_DB {

my($ptrLAYOUT,$thisQueryFieldName,$thisQueryKey,$LGDEBUG_FTN) = @_;

##print "** DSN = $LAYOUT{'thisMailingDSN'} \n";

my(%LAYOUT) = %$ptrLAYOUT;
my $RS = undef;
my $Conn = undef;
my $Errors = undef;
my @fieldNames = undef;
my $PID = -1;
my $recCount = 0;

# construct/open DSN

my $Conn = undef;

$Conn = new Win32::OLE("ADODB.Connection");
$Conn->Open($LAYOUT{'thisMailingDSN'},"","");

my(%DB) = undef;
$DB{'_exit_'} = 1;

# Load data into recordset

my $mailingTable = $LAYOUT{'thisMailingTable'};

$thisQuery = "SELECT * FROM $mailingTable WHERE $thisQueryFieldName = \'$thisQueryKey\'";

$RS = $Conn->Execute($thisQuery);

if(!$RS) {

        $Errors = $Conn->Errors();
        print "Errors:\n";
        foreach $error (keys %$Errors) {
                print $error->{Description}, "\n";
        }
        $eventType = 'FAIL'; $eventDesc = " \<".$thisRecordID."\>   ($LAYOUT{'thisMailingDSN'}) querySELECT_DB :: Cannot create RS: \[$thisQuery\]"; &logEvent(\*FH_log, $eventType, $eventDesc, $Guser, $LGDEBUG_FTN);
}

my $i = undef;

for($i=0; $i < $RS->Fields->Count; $i++) { $fieldNames[$i] = $RS->Fields->Item($i)->Name; }

my $RESULT_FLAG = 1;

if($RS->EOF) { $RESULT_FLAG = 0; }
$RS -> MoveNext;

if(!$RS->EOF) { $RESULT_FLAG = -1; }
$RS -> MovePrevious;

if($RESULT_FLAG < 1) {

        (($eventRequiresFUP =~ /badMDB/)?'':'.badMDB'); $FUPcount++; $eventType = 'ERR_FUP'; $eventDesc = " \<".$thisRecordID."\>   ($LAYOUT{'thisMailingDSN'}) querySELECT_DB :: Bad Lookup ($RESULT_FLAG): \[$thisQuery\]"; &logEvent(\*FH_log, $eventType, $eventDesc, $Guser, $LGDEBUG_FTN);

        $DB{'_exit_'} = $RESULT_FLAG;

return %DB;
}

while ( !$RS->EOF ) {

        for($i=0; $i < $RS ->Fields->Count; $i++) {

                $DB{$fieldNames[$i]}= uc($RS->Fields->Item($i)->value);
        }

        $RS->MoveNext; $recCount++;
}

$Conn->Close;
```

```
return %DB;
}

#
# --------------------------------------------------------------------
#

sub read_DSNtoAOH {

my($DSN,$LGDEBUG_FTN) = @_;

##print "** DSN = $DSN \n";

$RS = undef;
$Conn = undef;
$Errors = undef;
@fieldNames = undef;
$PID = -1;
$recCount = 0;

# construct/open DSN

$Conn = new Win32::OLE("ADODB.Connection");
$Conn->Open($DSN,"","");
my(@FILE) = undef;

# Load data into recordset

$thisQuery = "SELECT * FROM Email";

$RS = $Conn->Execute($thisQuery);

if(!$RS) {
        $Errors = $Conn->Errors();
        print "Errors:\n";
        foreach $error (keys %$Errors) {
                print $error->{Description}, "\n";
        }
        $eventType = 'FAIL'; $eventDesc = "read_DSNToAOH::Cannot create RS: \[$thisQuery\]"; &logEvent(\*FH_log, $eventType, $eventDesc, $Guser, $LGDEBUG_
FTN);
}

my $i = undef;

for($i=0; $i < $RS->Fields->Count; $i++) { $fieldNames[$i] = $RS->Fields->Item($i)->Name; }

while ( !$RS->EOF ) {

        for($i=0; $i < $RS ->Fields->Count; $i++) {

         $FILE[$recCount]{$fieldNames[$i]}= $RS->Fields->Item($i)->value;

        }

        $RS->MoveNext; $recCount++; if ($recCount % 20 == 0) { print STDOUT "+"; }
}

$Conn->Close;

return @FILE;
}

#
# --------------------------------------------------------------------
#

sub removeTabs {

my($tabs) = @_;
```

```

$tabs = s/\t//g;

return $tabless

}

#
# ----------------------------------------------------------------------
#

sub subjectSynonymnLookup {

my($subject,$ptrHASHLAYOUT) = @_;
my(%LAYOUT) = %$ptrHASHLAYOUT;
my($thisMailingID) = -1;

        $subject =~ /\s*([\=\*\+\-]{1,3})\s*/;
        $thisSubjectMailingIDToken = $1;

        if(defined($LAYOUT{$thisSubjectMailingIDToken})) { $thisMailingID = $LAYOUT{$thisSubjectMailingIDToken}; }

        ## print "-- debug -- ATTEMPT MAILING ID RECOVERY... TOKEN= |$thisSubjectMailingIDToken|... RESULT MAILINGID= |$thisMailingID|\n";

return $thisMailingID
}

#
# ----------------------------------------------------------------------
#

sub writeRecord {

# takes FH by ref; has _e_ feature for values

my($FH,$GfieldDelimiter,$logKey,$ptrHASHrecord,$RECORD_TYPE, $LGDEBUG_FTN) = @_;

flock($FH, $LOCK_EX);

my $fileEmpty = 0;
if(-z $FH) {$fileEmpty = 1;}

%HASHrecord = %$ptrHASHrecord;

$space = " ";
$exit = 1;

$LOCK_SH = 1;
$LOCK_EX = 2;
$LOCK_NB = 4;
$LOCK_UN = 8;

my $record = undef;
my $header = undef;
my $key = undef;

foreach $key(sort(keys(%HASHrecord))) {

        $value  = $HASHrecord{$key};

        if($fileEmpty) { $key =~ s/[a-z]{1,2}_//; $header .= (uc($key).$GfieldDelimiter); }

        if($value =~ /^_e_(.{2,})$/) { $value = eval($1); }


        $record .= ($value.$GfieldDelimiter);

}
chop($record);                                          # remove last field delimiter
```

```

if($fileEmpty) { chop($header); print $FH $header."\n"; }
if($ADD_CHANGE_ON && $RECORD_TYPE ne 'BODY') { $record = &canonicalize($record); }
print $FH $record."\n";

flock($FH, $LOCK_UN);
}

__END__

```

Claims

1. A machine-based method comprising

analyzing an e-mail message to derive response information concerning a commercial transaction, and based on the derived information, and

automatically generating commercial transaction data in a format that is usable to automatically complete the commercial transaction.

2. The method of claim 1 in which the commercial transaction comprises an order for a product or service.

3. The method of claim 1 in which the e-mail message comprises at least part of an e-mail sent to a customer and responses of the customer to the e-mail.

4. The method of claim 1 in which the automatic completion of the commercial transaction comprises order fulfillment.

5. A machine-based method comprising

sending an e-mail message to a customer offering a product or service for sale, the e-mail message comprising locations for response by the customer indicating his intention to order the product or service,

receiving from the customer an e-mail message that includes the response,

based on the received e-mail, automatically generating order information in a format usable automatically by an order fulfillment system to cause the order to be filled.

6. A machine-based method comprising

analyzing an e-mail message to derive response information concerning a commercial transaction,

automatically identifying response information which requires resolution of an issue with the source of the e-mail message, and automatically managing an e-mail dialog with the source to resolve the issue.

7. The method of claim 6 in which at least some of the e-mail dialog is performed automatically.

8. Software guided interactive e-mail dialogs to resolve, on behalf of a vendor, customer issues that occur in direct response e-mails that are automatically identified as requiring a dialog.

9. A machine-based method comprising

automatically sorting e-mail messages, based on response information contained in the messages, into e-mail messages that can be processed automatically to generate commercial transactions, e-mail messages in which the response information is inadequate to permit generation of commercial transactions, and e-mail messages that may be subjected to exception handling to yield information that is sufficient to generate commercial transactions.

10. A machine-based method comprising

analyzing an e-mail message to derive response information concerning a commercial transaction, and

automatically generating a confirmatory e-mail message to the source of the e-mail message confirming that the commercial transaction has been or will be completed.

11. A machine-based method comprising

receiving inbound e-mail messages that result from corresponding outbound e-mail messages associated with a marketing program, the inbound messages containing response information, each of the outbound messages being associated with a distinct piece of the marketing program, and

automatically associating the response information in each of the inbound messages with the corresponding distinct piece of the marketing program.

12. The method of claim 11 in which the piece comprises a marketing campaign or a marketing flight.

13. The method of claim 11 in which the inbound messages contain information that links them to the corresponding outbound messages, and the associating step uses the link information.

14. The method of claim 13 further comprising automatically parsing the inbound messages for order information.

15. A machine-based method comprising

sending outbound e-mail messages associated with commercial transactions,

storing information related to each of the outbound messages in a database, the information being useful for completing the commercial transactions, the information not being contained in the outbound messages,

analyzing inbound e-mail messages that result from the outbound messages and that contain response information useful in completing the commercial transactions, and

automatically merging the response information with corresponding information in the database for use in completing the transactions.

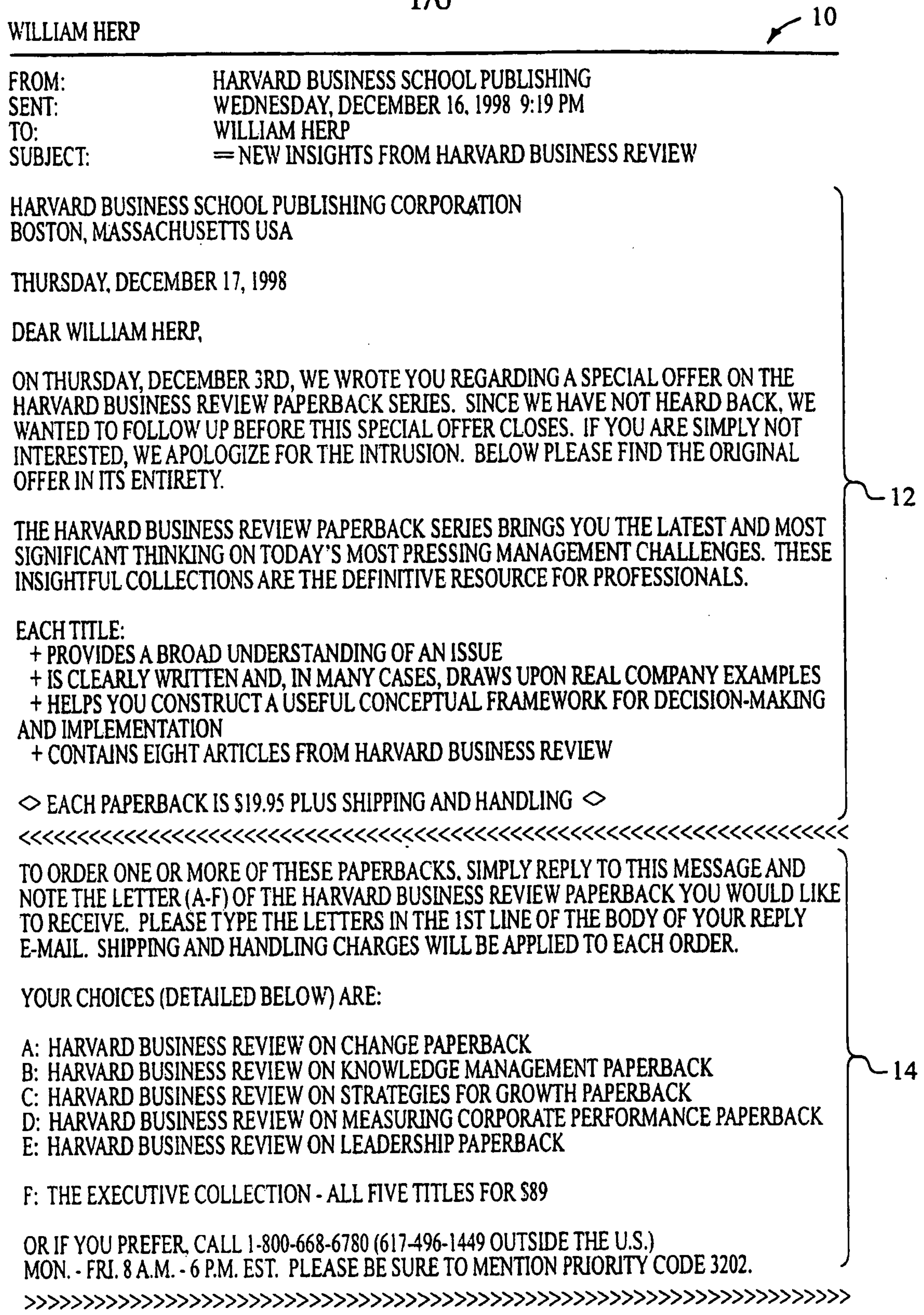
16. A machine-based method comprising

sending outbound e-mail messages associated with commercial transactions,

storing information related to each of the outbound messages in a database, the information being useful for completing the commercial transactions, the information not being contained in the outbound messages,

analyzing inbound e-mail messages that result from the outbound messages and that contain response information useful in completing the commercial transactions, identifying inbound e-mail messages that cannot be processed automatically to generate the commercial transactions, and

using the database information to assist in exception handling of the identified inbound messages.



WILLIAM HERP

FROM: HARVARD BUSINESS SCHOOL PUBLISHING
SENT: WEDNESDAY, DECEMBER 16, 1998 9:19 PM
TO: WILLIAM HERP
SUBJECT: = NEW INSIGHTS FROM HARVARD BUSINESS REVIEW

HARVARD BUSINESS SCHOOL PUBLISHING CORPORATION
BOSTON, MASSACHUSETTS USA

THURSDAY, DECEMBER 17, 1998

DEAR WILLIAM HERP,

ON THURSDAY, DECEMBER 3RD, WE WROTE YOU REGARDING A SPECIAL OFFER ON THE HARVARD BUSINESS REVIEW PAPERBACK SERIES. SINCE WE HAVE NOT HEARD BACK, WE WANTED TO FOLLOW UP BEFORE THIS SPECIAL OFFER CLOSES. IF YOU ARE SIMPLY NOT INTERESTED, WE APOLOGIZE FOR THE INTRUSION. BELOW PLEASE FIND THE ORIGINAL OFFER IN ITS ENTIRETY.

THE HARVARD BUSINESS REVIEW PAPERBACK SERIES BRINGS YOU THE LATEST AND MOST SIGNIFICANT THINKING ON TODAY'S MOST PRESSING MANAGEMENT CHALLENGES. THESE INSIGHTFUL COLLECTIONS ARE THE DEFINITIVE RESOURCE FOR PROFESSIONALS.

EACH TITLE:
+ PROVIDES A BROAD UNDERSTANDING OF AN ISSUE
+ IS CLEARLY WRITTEN AND, IN MANY CASES, DRAWS UPON REAL COMPANY EXAMPLES
+ HELPS YOU CONSTRUCT A USEFUL CONCEPTUAL FRAMEWORK FOR DECISION-MAKING AND IMPLEMENTATION
+ CONTAINS EIGHT ARTICLES FROM HARVARD BUSINESS REVIEW

◇ EACH PAPERBACK IS $19.95 PLUS SHIPPING AND HANDLING ◇

<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<

TO ORDER ONE OR MORE OF THESE PAPERBACKS, SIMPLY REPLY TO THIS MESSAGE AND NOTE THE LETTER (A-F) OF THE HARVARD BUSINESS REVIEW PAPERBACK YOU WOULD LIKE TO RECEIVE. PLEASE TYPE THE LETTERS IN THE 1ST LINE OF THE BODY OF YOUR REPLY E-MAIL. SHIPPING AND HANDLING CHARGES WILL BE APPLIED TO EACH ORDER.

YOUR CHOICES (DETAILED BELOW) ARE:

A: HARVARD BUSINESS REVIEW ON CHANGE PAPERBACK
B: HARVARD BUSINESS REVIEW ON KNOWLEDGE MANAGEMENT PAPERBACK
C: HARVARD BUSINESS REVIEW ON STRATEGIES FOR GROWTH PAPERBACK
D: HARVARD BUSINESS REVIEW ON MEASURING CORPORATE PERFORMANCE PAPERBACK
E: HARVARD BUSINESS REVIEW ON LEADERSHIP PAPERBACK

F: THE EXECUTIVE COLLECTION - ALL FIVE TITLES FOR $89

OR IF YOU PREFER, CALL 1-800-668-6780 (617-496-1449 OUTSIDE THE U.S.)
MON. - FRI. 8 A.M. - 6 P.M. EST. PLEASE BE SURE TO MENTION PRIORITY CODE 3202.

>>

FIG. 1A

A: ** HARVARD BUSINESS REVIEW ON CHANGE **
PROVIDES LANDMARK IDEAS TO HELP YOU UNDERSTAND THE BEST WAYS FOR YOUR ORGANIZATION TO MANAGE CHANGE. INCLUDES ARTICLES BY JOHN KOTTER AND MORE.
(240 PP/#8842/$19.95)

B. ** HARVARD BUSINESS REVIEW ON KNOWLEDGE MANAGEMENT **
HIGHLIGHTS THE LEADING-EDGE THINKING AND PRACTICAL APPLICATIONS ON HOW COMPANIES GENERATE, COMMUNICATE, AND LEVERAGE KNOWLEDGE ASSETS. INCLUDES ARTICLES BY PETER DRUCKER, JOHN SEELY BROWN, AND MORE.
(240 PP/#8818/$19.95)

C: ** HARVARD BUSINESS REVIEW ON STRATEGIES FOR GROWTH **
PRESENTS THE LATEST TACTICS FOR HELPING MANAGERS FIND AND EXPLOIT THE BEST OPPORTUNITIES FOR GROWTH AND PROFITABILITY. INCLUDES ARTICLES BY ARIE DE GEUS, JEFFREY RAYPORT, AND MORE.
(240 PP/#8850/$19.95)

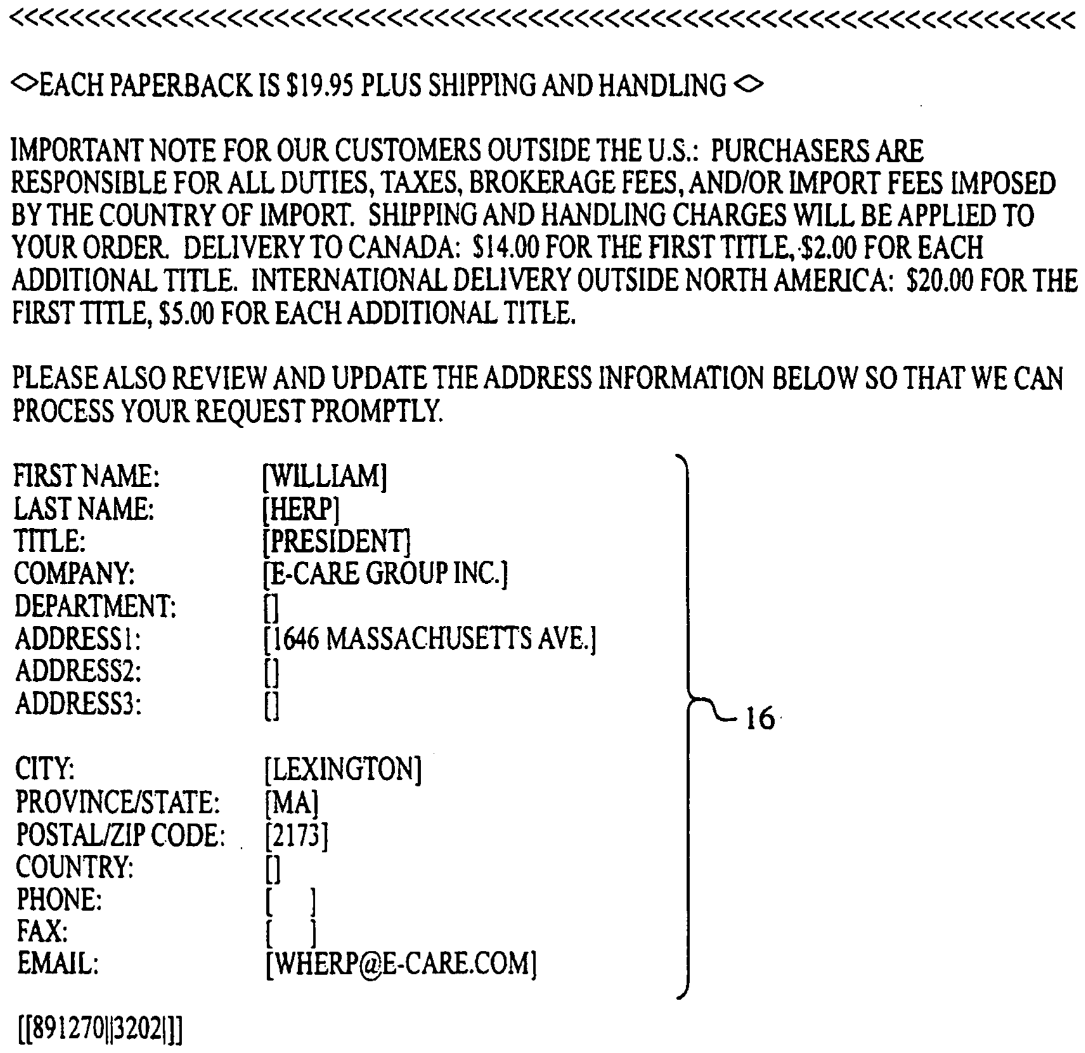
D: ** HARVARD BUSINESS REVIEW ON MEASURING CORPORATE PERFORMANCE **
OFFERS INSIGHT ON WHAT YOU NEED TO MEASURE AND HOW PERFORMANCE MEASURES CAN ALIGN AN ORGANIZATION AND BOOST PRODUCTIVITY. INCLUDES ARTICLES BY PETER DRUCKER, ROBERT KAPLAN AND DAVID NORTON, AND MORE.
(240 PP/#8826/$19.95)

E: ** HARVARD BUSINESS REVIEW ON LEADERSHIP **
PRESENTS PROVEN FUNDAMENTALS OF LEADERSHIP AND CHALLENGES MANY LONG-HELD ASSUMPTIONS ABOUT THE TRUE SOURCES OF POWER AND AUTHORITY. INCLUDES ARTICLES BY JOHN KOTTER, JOSEPH BADARACCO, JR., AND MORE.
(240 PP/#8834/$19.95)

F: PURCHASE THE EXECUTIVE COLLECTION (INCLUDES ALL FIVE TITLES) FOR JUST $89- A SAVINGS OF MORE THAN 10%. (PRODUCT #7419BN)

<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<

FIG. 1B



<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<

<>EACH PAPERBACK IS $19.95 PLUS SHIPPING AND HANDLING <>

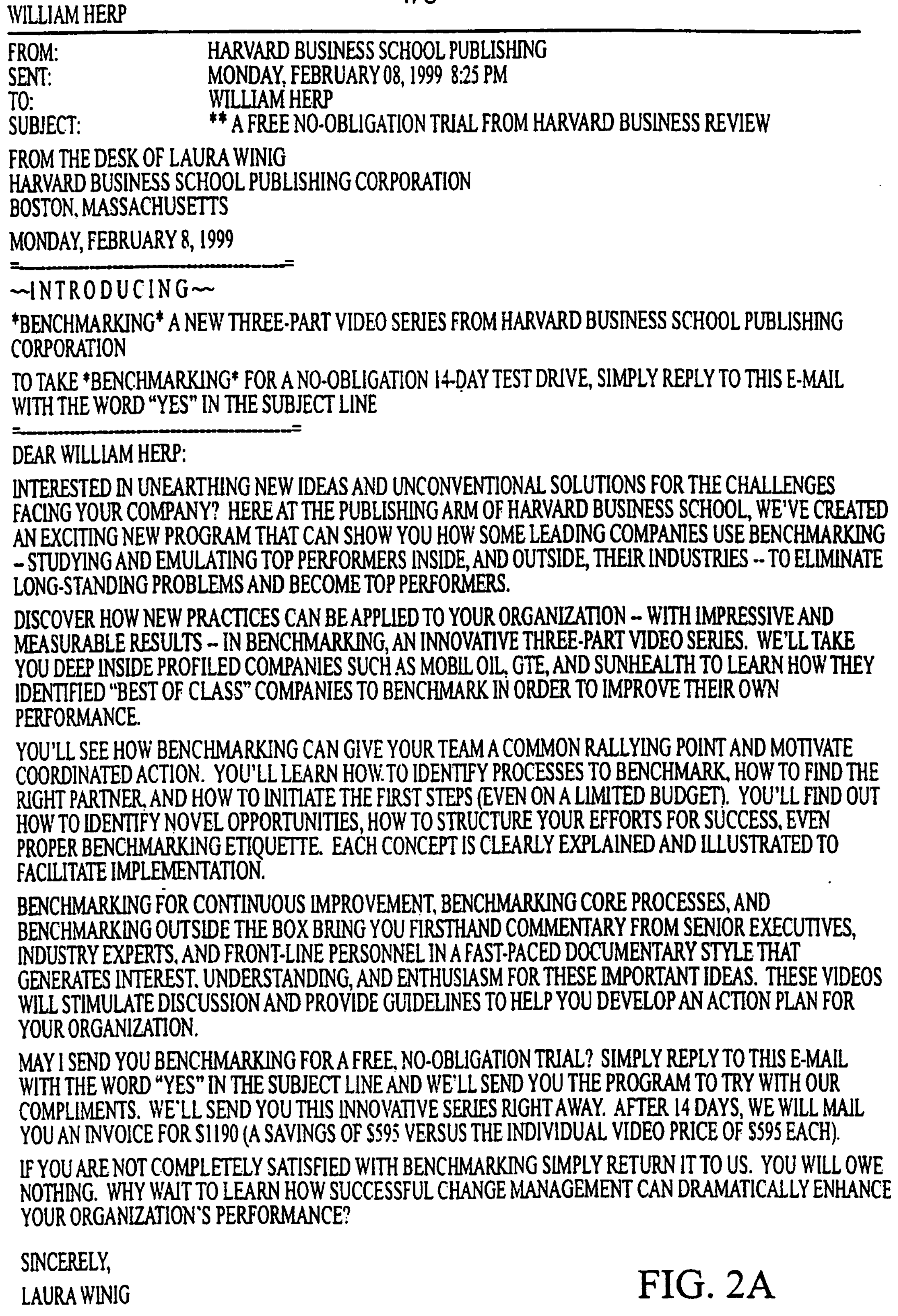
IMPORTANT NOTE FOR OUR CUSTOMERS OUTSIDE THE U.S.: PURCHASERS ARE RESPONSIBLE FOR ALL DUTIES, TAXES, BROKERAGE FEES, AND/OR IMPORT FEES IMPOSED BY THE COUNTRY OF IMPORT. SHIPPING AND HANDLING CHARGES WILL BE APPLIED TO YOUR ORDER. DELIVERY TO CANADA: $14.00 FOR THE FIRST TITLE, $2.00 FOR EACH ADDITIONAL TITLE. INTERNATIONAL DELIVERY OUTSIDE NORTH AMERICA: $20.00 FOR THE FIRST TITLE, $5.00 FOR EACH ADDITIONAL TITLE.

PLEASE ALSO REVIEW AND UPDATE THE ADDRESS INFORMATION BELOW SO THAT WE CAN PROCESS YOUR REQUEST PROMPTLY.

FIRST NAME:	[WILLIAM]
LAST NAME:	[HERP]
TITLE:	[PRESIDENT]
COMPANY:	[E-CARE GROUP INC.]
DEPARTMENT:	[]
ADDRESS1:	[1646 MASSACHUSETTS AVE.]
ADDRESS2:	[]
ADDRESS3:	[]
CITY:	[LEXINGTON]
PROVINCE/STATE:	[MA]
POSTAL/ZIP CODE:	[2173]
COUNTRY:	[]
PHONE:	[ ]
FAX:	[ ]
EMAIL:	[WHERP@E-CARE.COM]

[[891270||3202|]]

FIG. 1C



WILLIAM HERP

FROM: HARVARD BUSINESS SCHOOL PUBLISHING
SENT: MONDAY, FEBRUARY 08, 1999 8:25 PM
TO: WILLIAM HERP
SUBJECT: ** A FREE NO-OBLIGATION TRIAL FROM HARVARD BUSINESS REVIEW

FROM THE DESK OF LAURA WINIG
HARVARD BUSINESS SCHOOL PUBLISHING CORPORATION
BOSTON, MASSACHUSETTS

MONDAY, FEBRUARY 8, 1999

=-----------------------------------=

~INTRODUCING~

BENCHMARKING A NEW THREE-PART VIDEO SERIES FROM HARVARD BUSINESS SCHOOL PUBLISHING CORPORATION

TO TAKE *BENCHMARKING* FOR A NO-OBLIGATION 14-DAY TEST DRIVE, SIMPLY REPLY TO THIS E-MAIL WITH THE WORD "YES" IN THE SUBJECT LINE

=-----------------------------------=

DEAR WILLIAM HERP:

INTERESTED IN UNEARTHING NEW IDEAS AND UNCONVENTIONAL SOLUTIONS FOR THE CHALLENGES FACING YOUR COMPANY? HERE AT THE PUBLISHING ARM OF HARVARD BUSINESS SCHOOL, WE'VE CREATED AN EXCITING NEW PROGRAM THAT CAN SHOW YOU HOW SOME LEADING COMPANIES USE BENCHMARKING - STUDYING AND EMULATING TOP PERFORMERS INSIDE, AND OUTSIDE, THEIR INDUSTRIES -- TO ELIMINATE LONG-STANDING PROBLEMS AND BECOME TOP PERFORMERS.

DISCOVER HOW NEW PRACTICES CAN BE APPLIED TO YOUR ORGANIZATION -- WITH IMPRESSIVE AND MEASURABLE RESULTS -- IN BENCHMARKING, AN INNOVATIVE THREE-PART VIDEO SERIES. WE'LL TAKE YOU DEEP INSIDE PROFILED COMPANIES SUCH AS MOBIL OIL, GTE, AND SUNHEALTH TO LEARN HOW THEY IDENTIFIED "BEST OF CLASS" COMPANIES TO BENCHMARK IN ORDER TO IMPROVE THEIR OWN PERFORMANCE.

YOU'LL SEE HOW BENCHMARKING CAN GIVE YOUR TEAM A COMMON RALLYING POINT AND MOTIVATE COORDINATED ACTION. YOU'LL LEARN HOW TO IDENTIFY PROCESSES TO BENCHMARK, HOW TO FIND THE RIGHT PARTNER, AND HOW TO INITIATE THE FIRST STEPS (EVEN ON A LIMITED BUDGET). YOU'LL FIND OUT HOW TO IDENTIFY NOVEL OPPORTUNITIES, HOW TO STRUCTURE YOUR EFFORTS FOR SUCCESS, EVEN PROPER BENCHMARKING ETIQUETTE. EACH CONCEPT IS CLEARLY EXPLAINED AND ILLUSTRATED TO FACILITATE IMPLEMENTATION.

BENCHMARKING FOR CONTINUOUS IMPROVEMENT, BENCHMARKING CORE PROCESSES, AND BENCHMARKING OUTSIDE THE BOX BRING YOU FIRSTHAND COMMENTARY FROM SENIOR EXECUTIVES, INDUSTRY EXPERTS, AND FRONT-LINE PERSONNEL IN A FAST-PACED DOCUMENTARY STYLE THAT GENERATES INTEREST, UNDERSTANDING, AND ENTHUSIASM FOR THESE IMPORTANT IDEAS. THESE VIDEOS WILL STIMULATE DISCUSSION AND PROVIDE GUIDELINES TO HELP YOU DEVELOP AN ACTION PLAN FOR YOUR ORGANIZATION.

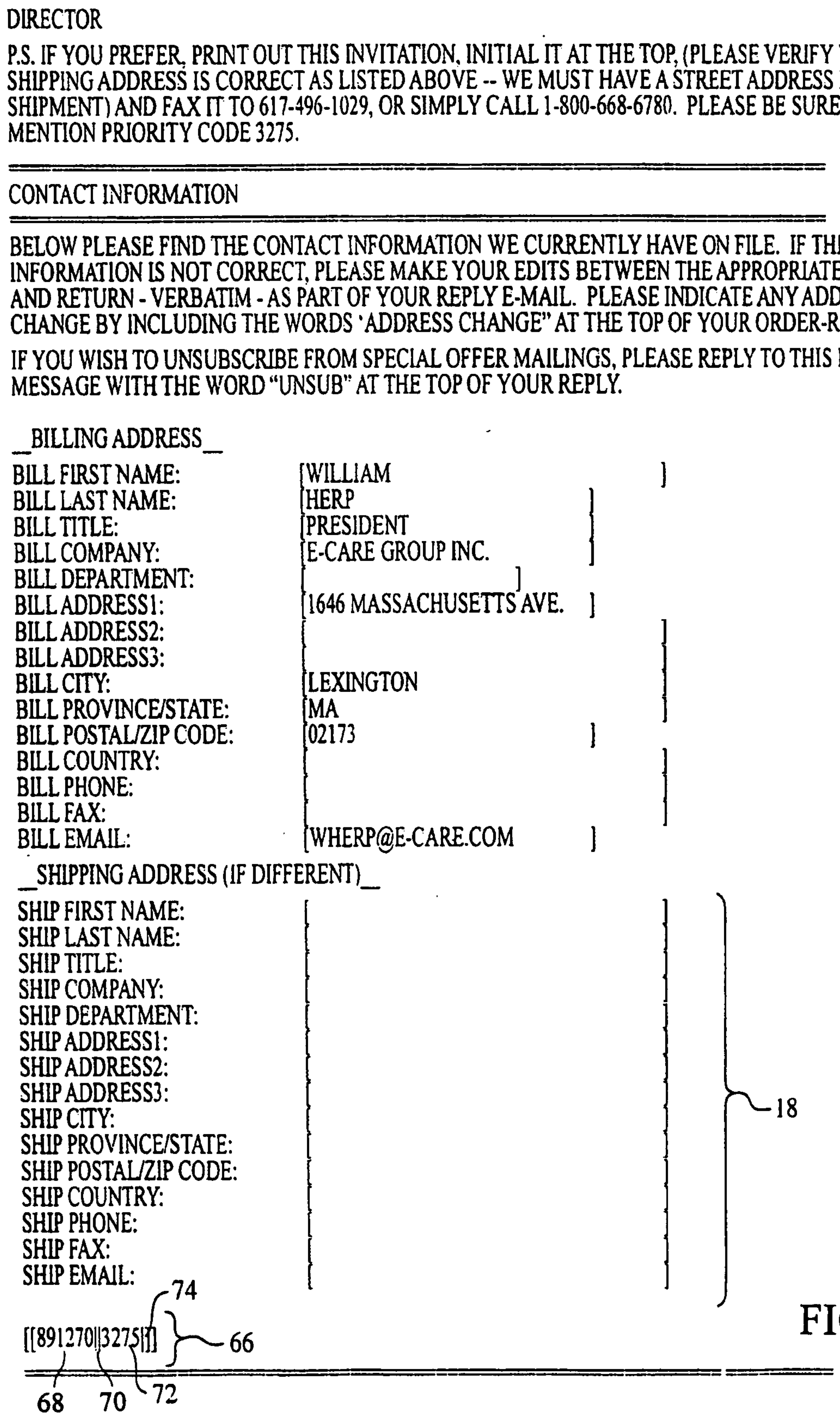
MAY I SEND YOU BENCHMARKING FOR A FREE, NO-OBLIGATION TRIAL? SIMPLY REPLY TO THIS E-MAIL WITH THE WORD "YES" IN THE SUBJECT LINE AND WE'LL SEND YOU THE PROGRAM TO TRY WITH OUR COMPLIMENTS. WE'LL SEND YOU THIS INNOVATIVE SERIES RIGHT AWAY. AFTER 14 DAYS, WE WILL MAIL YOU AN INVOICE FOR $1190 (A SAVINGS OF $595 VERSUS THE INDIVIDUAL VIDEO PRICE OF $595 EACH).

IF YOU ARE NOT COMPLETELY SATISFIED WITH BENCHMARKING SIMPLY RETURN IT TO US. YOU WILL OWE NOTHING. WHY WAIT TO LEARN HOW SUCCESSFUL CHANGE MANAGEMENT CAN DRAMATICALLY ENHANCE YOUR ORGANIZATION'S PERFORMANCE?

SINCERELY,

LAURA WINIG

FIG. 2A


DIRECTOR

P.S. IF YOU PREFER, PRINT OUT THIS INVITATION, INITIAL IT AT THE TOP, (PLEASE VERIFY YOUR SHIPPING ADDRESS IS CORRECT AS LISTED ABOVE -- WE MUST HAVE A STREET ADDRESS FOR SHIPMENT) AND FAX IT TO 617-496-1029, OR SIMPLY CALL 1-800-668-6780. PLEASE BE SURE TO MENTION PRIORITY CODE 3275.

CONTACT INFORMATION

BELOW PLEASE FIND THE CONTACT INFORMATION WE CURRENTLY HAVE ON FILE. IF THIS INFORMATION IS NOT CORRECT, PLEASE MAKE YOUR EDITS BETWEEN THE APPROPRIATE BRACKETS AND RETURN - VERBATIM - AS PART OF YOUR REPLY E-MAIL. PLEASE INDICATE ANY ADDRESS CHANGE BY INCLUDING THE WORDS 'ADDRESS CHANGE" AT THE TOP OF YOUR ORDER-REPLY.

IF YOU WISH TO UNSUBSCRIBE FROM SPECIAL OFFER MAILINGS, PLEASE REPLY TO THIS E-MAIL MESSAGE WITH THE WORD "UNSUB" AT THE TOP OF YOUR REPLY.

BILLING ADDRESS

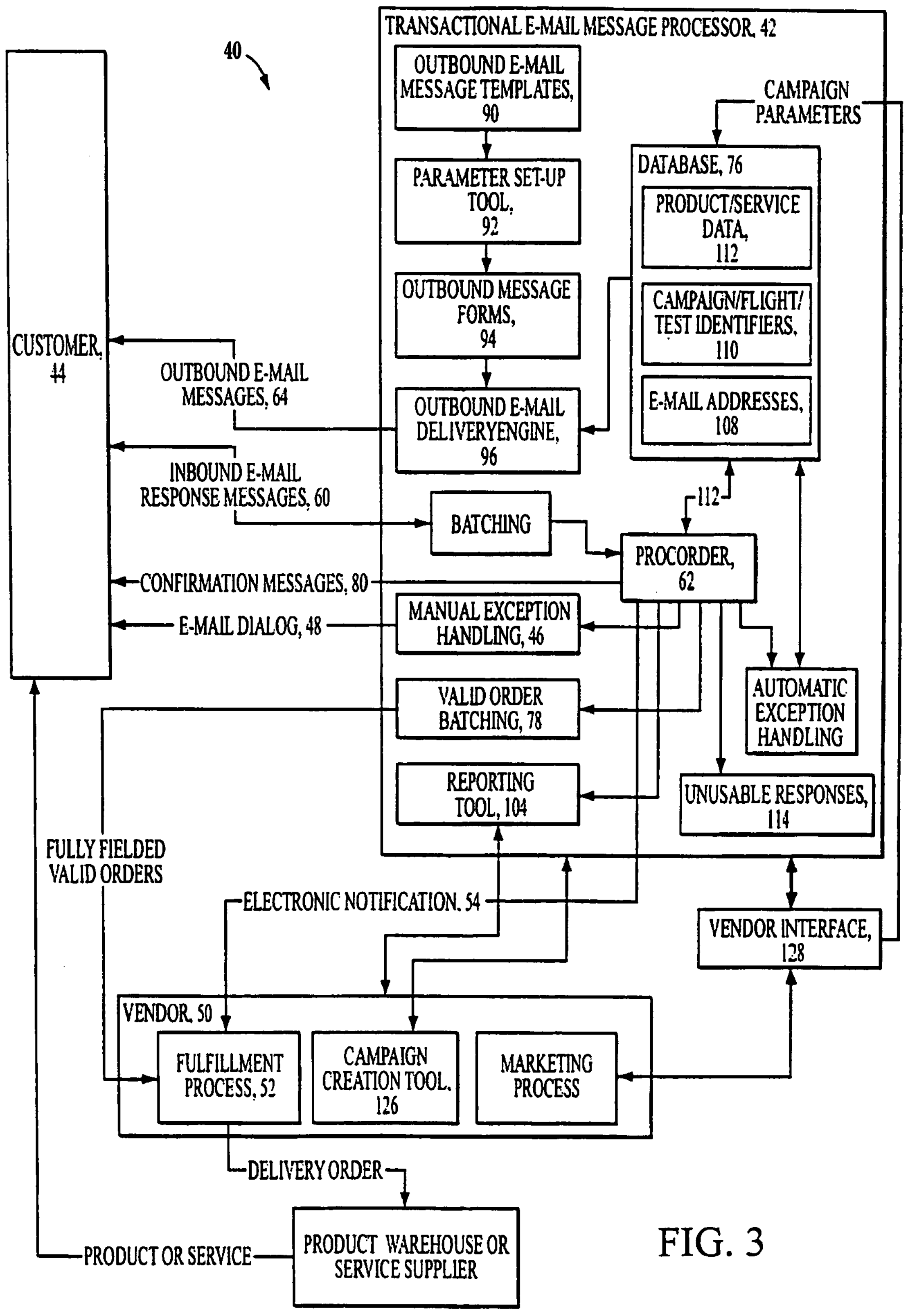
BILL FIRST NAME:	[WILLIAM]
BILL LAST NAME:	[HERP]
BILL TITLE:	[PRESIDENT]
BILL COMPANY:	[E-CARE GROUP INC.]
BILL DEPARTMENT:	[ ]
BILL ADDRESS1:	[1646 MASSACHUSETTS AVE.]
BILL ADDRESS2:	[ ]
BILL ADDRESS3:	[ ]
BILL CITY:	[LEXINGTON]
BILL PROVINCE/STATE:	[MA]
BILL POSTAL/ZIP CODE:	[02173]
BILL COUNTRY:	[ ]
BILL PHONE:	[ ]
BILL FAX:	[ ]
BILL EMAIL:	[WHERP@E-CARE.COM]

SHIPPING ADDRESS (IF DIFFERENT)

SHIP FIRST NAME:	[ ]
SHIP LAST NAME:	[ ]
SHIP TITLE:	[ ]
SHIP COMPANY:	[ ]
SHIP DEPARTMENT:	[ ]
SHIP ADDRESS1:	[ ]
SHIP ADDRESS2:	[ ]
SHIP ADDRESS3:	[ ]
SHIP CITY:	[ ]
SHIP PROVINCE/STATE:	[ ]
SHIP POSTAL/ZIP CODE:	[ ]
SHIP COUNTRY:	[ ]
SHIP PHONE:	[ ]
SHIP FAX:	[ ]
SHIP EMAIL:	[ ]

[[891270||3275||]]

FIG. 2B



40
TRANSACTIONAL E-MAIL MESSAGE PROCESSOR, 42
OUTBOUND E-MAIL MESSAGE TEMPLATES, 90
PARAMETER SET-UP TOOL, 92
OUTBOUND MESSAGE FORMS, 94
OUTBOUND E-MAIL DELIVERYENGINE, 96
CAMPAIGN PARAMETERS
DATABASE, 76
PRODUCT/SERVICE DATA, 112
CAMPAIGN/FLIGHT/ TEST IDENTIFIERS, 110
E-MAIL ADDRESSES, 108
CUSTOMER, 44
OUTBOUND E-MAIL MESSAGES, 64
INBOUND E-MAIL RESPONSE MESSAGES, 60
BATCHING
112
PROCORDER, 62
CONFIRMATION MESSAGES, 80
E-MAIL DIALOG, 48
MANUAL EXCEPTION HANDLING, 46
VALID ORDER BATCHING, 78
AUTOMATIC EXCEPTION HANDLING
REPORTING TOOL, 104
UNUSABLE RESPONSES, 114
FULLY FIELDED VALID ORDERS
ELECTRONIC NOTIFICATION, 54
VENDOR INTERFACE, 128
VENDOR, 50
FULFILLMENT PROCESS, 52
CAMPAIGN CREATION TOOL, 126
MARKETING PROCESS
DELIVERY ORDER
PRODUCT OR SERVICE
PRODUCT WAREHOUSE OR SERVICE SUPPLIER

FIG. 3

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US00/19403

A. CLASSIFICATION OF SUBJECT MATTER

IPC(7) :G06F 17/60
US CL : 705/1
According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

U.S. : 705/1,709/206,207,379/193,707/505

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

DIALOG, WEST,EAST

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	US 5,724,424 A (GIFFORD) 03 Mar 1998, Fig 1[68,64,66,67,62,200]; col 4 lines 48-53; col 5 lines 22-29; Fig.3[5,6,7]; col 5 lines 18-19;col 5 lines 34-37;Fig 5[18]; Fig 4[15]; col 5 lines 49-57; Fig 6[19-25];col 5 lines 29-44;Fig 4[8-17];Fig 2[1-3];Fig 3[3];col 5 line 25-col 6 line 5];Fig. 5;col 6 lines 20-29;col 6 lines 39-42;Fig.3[3]	1-16

☐ Further documents are listed in the continuation of Box C. ☐ See patent family annex.

*	Special categories of cited documents:	"T"	later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"A"	document defining the general state of the art which is not considered to be of particular relevance	"X"	document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"E"	earlier document published on or after the international filing date	"Y"	document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art
"L"	document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)	"&"	document member of the same patent family
"O"	document referring to an oral disclosure, use, exhibition or other means		
"P"	document published prior to the international filing date but later than the priority date claimed		

Date of the actual completion of the international search 19 AUGUST 2000	Date of mailing of the international search report 02 OCT 2000
Name and mailing address of the ISA/US Commissioner of Patents and Trademarks Box PCT Washington, D.C. 20231 Facsimile No. (703) 305-3230	Authorized officer Tod Swann [illegible] Telephone No. (703) [illegible]